University of Nottingham

School of Mathematical Sciences

Efficient Training and Implementation of Gaussian Process Potentials

Jack Broad

A thesis submitted to the University of Nottingham for the degree of

Doctor of Philosophy

July 2022

Contents

List of Figures

List of Tables

Abstract

Molecular simulations are a powerful tool for translating information about the intermolecular interactions within a system to thermophysical properties via statistical mechanics. However, the accuracy of any simulation is limited by the potentials that model the microscopic interactions. Most first principles methods are too computationally expensive for use at every time-step or cycle of a simulation, which require typically thousands of energy evaluations. Meanwhile, cheaper semi-empirical potentials give rise to only qualitatively accurate simulations. Consequently, methods for efficient first principles predictions in simulations are of interest.

Machine-learned potentials (MLPs) have shown promise in this area, offering first principles predictions at a fraction of the cost of *ab initio* calculation. Of particular interest are Gaussian process (GP) potentials, which achieve equivalent accuracy to other MLPs with smaller training sets. They therefore offer the best route to employing information from expensive *ab initio* calculations, for which building a large data set is time-consuming.

GP potentials, however, are among the most computationally intensive MLPs. Thus, they are far more costly to employ in simulations than semi-empirical potentials. This work addresses the computational expense of GP potentials by both reducing the training set size at a given accuracy and developing a method to invoke GP potentials efficiently for first principles prediction in simulations.

By varying the cross-over distance between the GP and a long-range function with the accuracy of the former, training by sequential design requires up to 40 % fewer training points at fixed accuracy. This method was applied successfully to the CO-Ne, HF-Ne, HF-Na^+, CO_2-Ne, $(CO)_2$, $(HF)_2$ and $(HCl)_2$ systems, and can be extended easily to other interactions and methods of prediction. Meanwhile, a significant reduction in the time taken for Monte Carlo displacement and volume change moves is achieved by parallelisation of the requisite GP calculations. Though this exploits in part the framework of GP regression, the distribution of the calculations themselves is general to other methods of prediction. The work also shows that current kernels and input transforms for modelling intermolecular interactions are not improved easily.

Acknowledgements

Firstly, I would like to express my unreserved gratitude to my supervisors, Richard Graham and Richard Wheatley. The constant guidance and encouragement with which you supplied me was beyond what I had any right to expect. You have both not only made this work possible but made undertaking it a great experience.

I would also like to thank my parents and grandparents, Debbie, Mark, Ivy and Bernie. Throughout my life you have all been sources of limitless support; your presence has always made me feel secure such that I could focus on the tasks in front of me and in that way you have all been instrumental in the production of the work laid out here.

And finally I would like to thank Tom and Cami. At different times and in different ways you both helped me through difficult situations. Though I am not sure either of you fully realised it, I hope my writing this shows in some small way my appreciation for the significance you both have in the development of this work.

Chapter 1

Introduction and overview

1.1 Introduction

Molecular simulations are useful in predicting properties of materials such as liquids for which first principles equations of state cannot be derived easily. In any simulation, approximations of the potential energy surfaces (PESs) that describe the relevant molecular interactions are a pre-requisite. Traditionally, these approximations have been made using semi-empirical potentials (force-fields) [1, 2]. An example of such a force-field is AMOEBA [3–5], which now includes polarisable three-body energies [6, 7]. This force-field has been applied recently to simulations of a variety of systems, from nucleic acids [8] to protein-metal interactions [9, 10].

However, AMOEBA models the interactions in the system via the multipole expansion, which does not account for short-range repulsion between electrons. This effect is instead accounted for empirically using a dampening function. Its predictions are therefore not of the quality of those derived from first principles. In general, force-fields are limited by closed functional forms that require fitted parameters. Such forms restrict their capacity to capture the complicated topography of the PES. Furthermore, force-fields fitted to *ab initio* or experimental data are laborious to produce and may fail to capture accurately even the fitted data. As the accuracy of a simulation depends on the potential employed, much work has been devoted to developing force-fields that provide approximations of PESs with quantum-mechanical accuracy. Because quantum mechanics can be used to derive first principles predictions of the intermolecular interaction energies, such potentials would achieve first principles accuracy and facilitate quantitatively accurate simulations. These simulations would eliminate the need for experiment when assessing, for example, phase coexistence in fluids. This is of particular interest where experiments are dangerous, laborious, expensive or inaccurate.

One method to invoke first principles predictions in simulations is to generate *ab initio* force-fields [11]. These derive functional forms for the potential from first principles and work by treating intermolecular interaction energies as perturbations of the individual molecular energies. However, many of these force-fields are not based entirely on first principles. For example, the effective fragment potential [12] and sum of interaction between fragments *ab initio* potential [13] contain empirical terms that must be fitted to experimental or *ab initio* data.

The work presented here pertains to an alternative approach: attempts to 'learn' the PES via a machine learning algorithm [14, 15]. Supervised machine learning algorithms, which have also been applied in other fields in chemistry and materials science [16–18], update their parameters using data. In the context of machine-learned potentials (MLPs), the approach proceeds by training a statistical technique on a relatively small set of data from quantum-chemical calculations on the PES of interest. This small data set is known as the training set. Many techniques have been employed to predict the energy in these algorithms, including neural networks [19–23], moment tensors [24–26] and Gaussian processes (GPs) [27–45]. MLPs offer a route to employ first principles predictions in simulations at massively reduced cost compared with using *ab initio* methods directly.

The first MLPs invoked neural networks (NNs) [46] as the method of prediction. NN potentials were then used to model the high-dimensional PES of Si in both solid and liquid states [19] before applications to systems of more than one element such as ZnO_2 [20] and ZnO_2 with Cu clusters [21]. These NN potentials modelled the energy at a given atom as the sum of the energies of the interatomic interactions within a pre-ordained cross-over distance, R_{cross}, of 6 Å. In systems of more than one element, interactions beyond R_{cross} were approximated as the energy of the electrostatic interactions between the atoms. This accounted for potential charge transfer between different atomic species [20]. Meanwhile, for systems comprising only one element, the energy of interaction above R_{cross} was assumed to be zero [19].

These NN potentials achieved promising results. The energy versus volume curves produced for a number of Si crystal structures were in good agreement with the data from density functional theory (DFT) [19], while the NN-predicted energies of various Zn_4O_4 clusters were found to match closely those from DFT [20]. However, it has since been found that Gaussian Processes [47, 48] have the capacity to outperform NNs in terms of predictive accuracy at a fixed training set size [31, 49, 50]. This advantage, though, comes with a larger computational cost of prediction for a given size of training set [31, 49, 50]. The disparity in cost between GP potentials and traditional force-fields is yet larger, meaning reducing the amount of training data required for a GP potential is paramount. The smaller training set sizes for GPs make them ideal candidates for transfer learning [51, 52], which in this context entails upgrading the training set energies to a more accurate *ab initio* method once they have been selected from a reference set. GP potentials, therefore, represent an excellent route to apply the most accurate first principles predictions to simulations.

Resultantly, all work herein is concerned with GP potentials. An existing example of a force-field that uses GP potentials is FFLUX [53], which has been used to approximate the energies of weakly bound complexes [54] and water clusters [55], among other applications [56–58]. Furthermore, in the field of materials science, GP models that invoke a smooth overlap of atomic positions (SOAP) kernel [59] have seen successful applications to many systems [40–42, 60–63].

GP potentials have also produced promising results in applications to intermolecular interactions [27–30]. Initially [27], the training sets for these models were constructed with Latin hypercube sampling [64–66]. Sequential design strategies [67,68], which achieve a prescribed accuracy with a smaller training set, have since been shown to outperform such methods [29]. Regardless of training set design, R_{cross} was imposed prior to training, just as for the NN potentials. This parameter defined a boundary beyond which a simple, long-range asymptotic function took over prediction from the GP [27, 29]. A similar approach is used in materials science, in which the contribution by one atom to the neighbour density of another was assumed to be zero if the two were separated by a distance in excess of R_{cross} [42,62]. Cross-over distances have been applied alongside other statistical methods of prediction too [15, 50], with moment tensor [25] potentials employing fixed, pre-determined cross-over distances. These distances are applied as they prevent the use of data to fit the long-range portion of the PES, where the energy tends asymptotically to zero.

Methods to achieve linear scaling for density functional theory (DFT) calculations have been produced [69, 70], offering an alternative to MLPs. However, said scaling is linear in the number of atoms in the simulation, which will exceed the size of a sequentially-designed training set. This is significant as the computational cost of prediction with GPs scales linearly with the size of this set. Moreover, for systems where dispersion interactions comprise a significant portion of the intermolecular energy DFT is a poor choice as it does not approximate these interactions from first principles.

Thus MLPs offer a more flexible route to apply *ab initio*-quality potentials because they in general allow the same algorithm to be used for predicting PESs of various chemical systems. Moreover, they allow the quality of the training data to be easily modified. GP potentials are particularly promising due to their suitability to transfer learning, which renders them ideal for applying high-level first principles predictions to simulations. A possible application of these potentials is to the operation of carbon capture and storage (CCS) pipelines, which transport impure CO_2 mixtures at pressures near to the critical point to maximise flow rate.

However, the exact temperature and pressure required to reach the critical point for different mixtures varies based on their composition. Re-creating this composition exactly to parameterise force-fields for simulations is difficult, but without such force-fields only qualitatively accurate estimates of the phase boundary are possible. Furthermore, dispersion interactions will form a significant contribution to the energy, rendering DFT unsuitable.

Applying GP potentials to simulations of CCS pipelines represents a possible solution: build potentials for each interaction between the small molecules in the mixture and apply these to the simulation. Using first principles predictions from these potentials will result in a simulation that itself achieves first principles accuracy, which will permit the various thermophysical properties that affect transport through CCS pipelines to be determined for different mixtures without experiment. As the compositions of the mixtures vary greatly between pipelines, modelling these properties would otherwise require extensive data from laborious experiment. Owing to the expected importance of simulations in CCS operations [71], GP potentials could have an important role in combating climate change. For example, the UK government has said that CCS "has the potential to decarbonise the economy and maximise economic opportunities for the UK" [72] once sufficient cost reductions are made to its deployment and safety concerns are addressed. These are processes in which GP potentials may prove instrumental.

Currently, the derivation of virial coefficients [73] to develop an equation of state (EoS) is a common method for determining first principles thermophysical properties of gases. Recently, these have been developed for helium [74] and used to determine several properties of pure ethane [75]. Virial EoS are produced relatively easily by integrating an appropriate PES. However, CCS pipelines commonly contain N_2, O_2, Ar and H_2 in addition to CO_2 [28]. In such cases, virial EoS are harder to produce due to the variety of two- and three-body interactions. This, though, is mitigated by the use of GPs, which allow PES models to be found straightforwardly. All possible combinations of two- and three-body interactions are needed for a virial EoS, which accounts for all possible two-body interactions in its second coefficient, all possible three-body terms in its third coefficient and so on. Higher-order coefficients are often negligible, however.

Despite their utility in the gas phase, virial EoS are not produced readily for liquids. Though empirical EoS can be developed in the liquid phase, these are inaccurate for mixtures of different compositions to those used in fitting and so require extensive experimental data. In addition, they do not offer first principles accuracy. Hence the use of molecular simulations for CCS pipelines is preferred. Use of a GP potential in such a simulation would permit the derivation of a first principles liquid EoS via a method to systematically derive such an equation from the simulation [76, 77].

Improvements to the training and implementation of GP potentials are significant for two reasons: MLPs are in general more computationally intensive than force-fields, and GP potentials are among the most expensive MLPs. The aforementioned eligibility of GPs in transfer learning means their efficient implementation offers the best route to applying high-level (e.g. coupled cluster) *ab initio* information in simulations. This is because minimising the number of these expensive calculations is also important from a computational standpoint. Methods for the efficient training and implementation of GP potentials are therefore of great interest. Holistically, this work details strategies that could improve vastly the accuracy of computationally tractable molecular simulations by reducing the training set size of GP potentials and implementing them in simulations efficiently.

This is achieved via improvements to the methodology employed to train and implement GP potentials. Regarding training, this work postulates that GP potentials can be developed more efficiently if the value of R_{cross} is not fixed but learned from the reference data. This method is explored in Chapter 3 and is applicable to other methods of prediction or modelling problems. The effect of alternative kernel functions and transforms on the inputs and outputs on training efficiency is then interrogated in Chapter 4. This was undertaken with the expectation that enhancing the stationarity of the data or making the kernel more robust to non-stationarity would improve training outcomes for certain systems. The work also posits that the application of GP potentials to simulations can be made more efficient via parallelisation, which is discussed in Chapter 5. Parallel calculations pervade many branches of science, which is reflected in their breadth of applications ranging from Monte Carlo simulation of photon transport for medical applications [78] to simulation of the fragmentation of soil following an explosion [79]. More recently, a parallel program for executing lattice-Boltzmann simulations was developed [80], which was designed to work across varied computational set-ups. This follows the principles of map-reduce parallel programming [81], which aims to permit automatic optimisation of parallel code across various devices.

1.2 Overview of structure

This work begins with a discussion of relevant background information in Chapter 2, which includes a discussion of GPs, how they make predictions and the root of their large computational cost. This precedes an overview of PESs and *ab initio* methods, before a discussion of intermolecular interactions. The former outlines what PESs are and how a range of *ab initio* methods work. The overview of intermolecular interactions includes common interaction types, as well as a brief outline of the multipole expansion. Finally, statistical mechanics and molecular simulations are considered in order to motivate the work shown in Chapter 5.

Chapters 3, 4 and 5 are the chapters containing original research. The results in Chapter 4 show that more complicated transformations on the inputs and outputs, and more complex kernel functions do not significantly affect the efficiency of training. The results in Chapters 3 and 5, meanwhile, contribute directly to the stated aims of increasing the efficiency of training and implementation of GPs. The first of these chapters details the methods for optimising R_{cross}, leading to an up to 40 % reduction in training set size for fixed accuracy. The latter explores implementing GP potentials in simulations using parallel programming to exploit that the GP predictions are a product of nested sums over exponentials. All of these exponentials can be pre-computed and the nested sums parallelised. This leads to a nearly five-fold reduction in computational time for a non-additive energy calculation over five processes and a 15-fold reduction over 40 processes.

1.3 List of publications

The work in Chapter 3 has already been published. Other than my supervisors, I was the only person working on this project and am responsible for the work in its entirety. The same is true of both chapters said work precedes. Though the work in Chapter 4 will not be published, the work in Chapter 5 will be suitable once the modifications in the future work section are added.

Chapter 2

Background theory

2.1 Gaussian processes

Gaussian processes (GPs) [47] are non-parametric statistical methods of prediction that can be employed in both regression and classification tasks. This work focuses on the former, where GPs have seen recent applications ranging from the estimation of the critical temperature of an Fe-based superconductor [82] to the solution of partial differential equations [83]. The prevalence of GPs is highlighted further by the development of deep GPs [84], which share the same structure as neural networks but with a GP at each hidden layer. This method exploits the fact that a GP is equivalent to a neural network with a single hidden layer comprising an infinite number of nodes [85]. Deep GPs are becoming so commonplace that a library for their implementation was recently developed for Python [86].

In the context of models of potential energy surfaces, the ability of GPs to interpolate multi-dimensional functions is paramount. This property allows them to capture the underlying potential energy surface from a relatively small number of *ab initio* calculations. This means that GPs are well-suited to the development of potentials that produce first principles predictions. Their success in doing so has been illustrated in a variety of chemical contexts, from intermolecular interactions [27,29] to solid-state interactions [87]. It has also been shown that the vibrational spectrum of H_2CO is predicted with more accuracy and less variability between fits by a GP when compared with a neural network [49]. GP potentials in addition achieve the lowest error per atom for simulations of various metals when compared with a variety of machine learning methods [50]. Furthermore, GPs were shown to predict the energies of $(H_2O)_x$ clusters with greater accuracy than neural networks for $x \geq 3$ [31]. In all of these examples, however, the computational cost of prediction of the GP exceeded that of the other methods. As this cost scales linearly with training set size, methods to reduce this set for GP potentials are of interest.

This section introduces GPs and GP regression to describe their use in modelling potential energy surfaces. The main purpose is to address what GPs are; how they make predictions; why prediction and re-optimisation of Gaussian processes is so computationally intensive; what different kernel functions are available; and how GP potentials were developed previously. None of these sections are exhaustive, with more detail available in the given references, but they aim to introduce these topics to facilitate and frame subsequent discussions.

This section draws heavily on 'Gaussian Processes for Machine Learning' by Rasmussen and Williams [47], specifically chapters two, four and five. Additional information was also taken from other sources [88,89] where required. Meanwhile, subsection 2.1.4 describes briefly the work of Uteva *et al.* [27, 29], providing an overview of how Gaussian process models of potential energy surfaces were trained and the results obtained from this training.

2.1.1 An introduction to Gaussian processes

Any function, $f(x) \in \mathbb{R}$, comprises an infinite number of points whether $x \in [0, 1]$ or $x \in [-\infty, \infty]$. As such, $f(x)$ is described by an infinite dimensional distribution. Given a finite number of observations N_t, the finite dimensional distribution required to approximate the underlying distribution of $f(x)$ can be inferred [89]. When approximating $f(x)$ as a Gaussian process (GP), the finite dimensional distribution is assumed to be multivariate Gaussian with N_t dimensions.

This work is concerned with the approximation of potential energy surfaces. These are discussed in detail in Section 2.2, but for now it is sufficient to consider a potential energy surface as a multivariate function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^{N_\mathrm{d}}$, with $\mathbf{x}$ a set of coordinates describing a point on the multidimensional surface and N_d the length of $\mathbf{x}$. Once more, only N_t instances of $\mathbf{x}$ need be observed to approximate

$f(\mathbf{x})$ as a GP. The set $\mathcal{T} = \{\mathbf{x}_i, \mathrm{Y}_i\}_{i=1}^{N_\mathrm{t}}$ is referred to henceforth as the training set for the GP, where Y_i is the observed energy at point $\mathbf{x}_i$ and N_t is the number of training points.

When approximating $f(\mathbf{x})$, it is assumed that the distribution describing $f(\mathbf{x})$ is a Gaussian process for which the finite dimensional distribution is multivariate Gaussian distributed. This is denoted

$$f(\mathbf{x}) \sim \mathcal{GP}(\boldsymbol{\mu}, \overline{\mathbf{K}}), \tag{2.1}$$

where $\overline{\mathbf{K}}$ is a matrix containing the covariances between the values of $f(\mathbf{x})$ at all observed $\mathbf{x}$ and $\boldsymbol{\mu}$ is the mean vector,

$$\boldsymbol{\mu} = (\mu_1, ..., \mu_{N_\mathrm{t}}), \tag{2.2}$$

of length N_t. This is filled using a mean function $\mu(\mathbf{x}_i) = \mu_i$. It is common to specify that $\mu(\mathbf{x}_i) = 0 \ \forall \ i$, leaving the GP predictions to tend to zero away from the observations. This is reasonable for potential energy surfaces, where the long-range energies tend asymptotically towards zero at large separations, and is therefore the case here.

Meanwhile, the N_t x N_t positive-definite covariance matrix is

$$\overline{\mathbf{K}} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_{N_\mathrm{t}}) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_{N\mathrm{t}}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_{N_\mathrm{t}}, \mathbf{x}_1) & k(\mathbf{x}_{N_\mathrm{t}}, \mathbf{x}_2) & \cdots & k(\mathbf{x}_{N_\mathrm{t}}, \mathbf{x}_{N_\mathrm{t}}) \end{bmatrix}. \tag{2.3}$$

This shows the covariance between all training points in its off-diagonal and the variances of each along its diagonal. $\overline{\mathbf{K}}$ is filled using a covariance (or kernel) function $k(\mathbf{x}_i, \mathbf{x}_j)$. Different types of kernel function are considered in section 2.1.3.

Fitting a GP can be interpreted in terms of Bayesian inference, which exploits Bayes' theorem,

$$p(M|D) = \frac{p(D|M)p(M)}{p(D)}. \tag{2.4}$$

Equation 2.4 is derived straightforwardly from the law of conditional probability,

$$p(M|D) = \frac{p(M \cap D)}{p(D)}. \tag{2.5}$$

In these equations, M represents a model that is fitted to some data D. $p(M|D)$ is the posterior distribution, $p(D|M)$ is the likelihood $\mathcal{L}$, $p(M)$ is the prior and $p(D)$ is the marginalisation constant. $p(M \cap D)$ in equation 2.5 is the joint distribution of M and D.

The quality of the fit to the data in the posterior distribution is determined by the hyperparameters[1] $\boldsymbol{\theta}$ of the kernel function. As $P(M|D) \propto \mathcal{L}$, optimisation of the hyperparameters is undertaken by maximising

$$\mathcal{L} = \prod_{i}^{N_\mathrm{t}} p_i(\mathrm{Y}_i|\boldsymbol{\theta}). \tag{2.6}$$

It is assumed that the data are independent and identically distributed. Assuming also a multivariate Gaussian finite dimensional distribution,

$$p_i(\mathrm{Y}_i|\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi|\overline{\mathbf{K}}|}} \exp\Big(-\frac{1}{2}(\mathrm{Y}_i - \mu_i)^\mathrm{T}\overline{\mathbf{K}}^{-1}(\mathrm{Y}_i - \mu_i)\Big). \tag{2.7}$$

Given a zero mean function, $\boldsymbol{\theta}$ contains the hyperparameters of the kernel function only and substituting equation (2.7) into equation (2.6) yields

$$\mathcal{L} = (2\pi)^{-\frac{N_\mathrm{t}}{2}}|\overline{\mathbf{K}}|^{-\frac{1}{2}} \exp\Big(-\frac{1}{2}\mathbf{Y}^\mathrm{T}\overline{\mathbf{K}}^{-1}\mathbf{Y}\Big). \tag{2.8}$$

Here, $|\overline{\mathbf{K}}|$ is the determinant of the covariance matrix and $\mathbf{Y}$ is a vector of observed energies.

Rather than equation (2.8), it is simpler to maximise the log-likelihood,

$$\log(\mathcal{L}) = -\frac{N_\mathrm{t}}{2}\log(2\pi) - \frac{1}{2}\log|\overline{\mathbf{K}}| - \frac{1}{2}\mathbf{Y}^\mathrm{T}\overline{\mathbf{K}}^{-1}\mathbf{Y}, \tag{2.9}$$

where "log" denotes the natural logarithm. This equation demonstrates that optimisation of the kernel function requires inversion of the N_t x N_t covariance matrix. Though the symmetry of the matrix can be exploited to increase the speed of this inversion using, for example, the Cholesky decomposition [90–92] (Appendix A), it still scales as order $\mathcal{O}(N_\mathrm{t}^3)$ [47,93]. Consequently, it is of interest to produce the smallest possible GP training sets to maximise the speed with which $\overline{\mathbf{K}}^{-1}$ can be calculated. This is the first of several reasons to minimise N_t that are introduced in this section.

[1]The use of this term rather than "parameters" is explained in subsection 2.1.3.

2.1.2 Gaussian process regression

After an arbitrary number of training observations N_t, a prediction at a new point $\mathbf{x}_*$ of the value Y_* can be made via GP regression. Assuming the training data in $\mathcal{T}$ are subject to noise σ_n^2, the joint distribution between the unobserved point and these data is

$$\begin{bmatrix} \mathbf{Y} \\ \mathrm{Y}_* \end{bmatrix} \sim \mathcal{N}\Big(0, \begin{bmatrix} \overline{\mathbf{K}} + \bar{\mathbf{I}}\sigma_\mathrm{n}^2 & \mathbf{K}_*^\mathrm{T} \\ \mathbf{K}_* & k(\mathrm{x}_*, \mathrm{x}_*) \end{bmatrix}\Big). \tag{2.10}$$

Here, $\bar{\mathbf{I}}$ is the identity matrix and

$$\mathbf{K}_* = (k(\mathbf{x}_*, \mathbf{x}_1), ..., k(\mathbf{x}_*, \mathbf{x}_{N_\mathrm{t}})) \tag{2.11}$$

is a vector that contains the covariance between $\mathbf{x}_*$ and all training points.

The posterior (*i.e.* predictive) distribution is obtained by conditioning the joint distribution on the training data (see Appendix B), yielding

$$\mathrm{Y}_*|\mathrm{x}_*, \mathcal{T} \sim \mathcal{N}(\mathbf{K}_*^\mathrm{T}(\overline{\mathbf{K}} + \bar{\mathbf{I}}\sigma_\mathrm{n}^2)^{-1}\mathbf{Y}, k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_*^\mathrm{T}(\overline{\mathbf{K}} + \bar{\mathbf{I}}\sigma_\mathrm{n}^2)^{-1}\mathbf{K}_*). \tag{2.12}$$

Therefore the predicted value of Y_* is

$$\mathrm{Y}_* = \mathbf{K}_*^\mathrm{T}(\overline{\mathbf{K}} + \bar{\mathbf{I}}\sigma_\mathrm{n}^2)^{-1}\mathbf{Y} \tag{2.13}$$

and the uncertainty in that prediction $var(\mathrm{Y}_*)$ is

$$var(\mathrm{Y}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_*^\mathrm{T}(\overline{\mathbf{K}} + \bar{\mathbf{I}}\sigma_\mathrm{n}^2)^{-1}\mathbf{K}_*. \tag{2.14}$$

It is a natural advantage of GPs that any prediction has a concomitant estimate of the uncertainty.

Equation (2.13) implies that predictions from GPs also incur a significant computational cost, as they entail inversion of $(\overline{\mathbf{K}}+\bar{\mathbf{I}}\sigma_\mathrm{n}^2)$. However, $(\overline{\mathbf{K}}+\bar{\mathbf{I}}\sigma_\mathrm{n}^2)^{-1}\mathbf{Y}$, known as the Woodbury vector, has to be determined once only for a fixed $\mathcal{T}$. This means that the computational cost of prediction scales as $\mathcal{O}(N_\mathrm{t})$ if no information on the variance is desired. Though this is vastly better than the scaling of the cost of re-optimisation, it can still become obstructively expensive for large training sets and means that GP potentials are more computationally intensive than force-fields. This shows once again that minimising the size of the training set when using GPs is paramount.

Neural network (NN) potentials also have a lower associated cost of prediction than GP potentials [31, 49, 50], though, as mentioned, GP potentials exhibit greater predictive accuracy in a variety of contexts. For a single-layer NN comprising N_{nodes} nodes, the cost of prediction at $\mathbf{x}_*$ (of length N_{d}) is dominated by the matrix-vector multiplication $\overline{\mathbf{W}}\mathbf{x}_*$. Here $\overline{\mathbf{W}}$ is an N_{nodes} x N_{d} matrix of weights from training [49]. This involves no matrix inversion, meaning its computational cost of prediction is smaller than that required of a GP. However, as GP potentials achieve comparable accuracy to NN potentials with fewer training points this disadvantage can be mitigated by an intelligent training strategy. This property also means that when "it is important to minimise the number of *ab initio* points, GP regression is a better choice" [49], hence the earlier assertion that GP potentials are ideal among machine-learned potentials for transfer learning.

2.1.3 Kernel functions

The covariance matrix that is inverted during GP regression is filled by a kernel function, meaning this function is central in predictions from GP potentials. Kernel functions are so named as they map the problem being modelled from the input space to an N_{t}-dimensional feature space. When modelling potential energy surfaces, the input space comprises the coordinates that describe specific points on the surface, while the feature space is the covariance between those points.

As shown, maximisation of the log-likelihood is used to optimise the hyperparameters of the kernel function based on the training data. The hyperparameters of the kernel function are so named because they do not parameterise directly an approximation of the function being modelled. They instead control the covariance between different points on the function, which is in turn used to make predictions of the function value. Put simply, the similarity in feature space between the training points and an unobserved point defines the prediction at the latter, and the hyperparameters of the kernel function define that similarity.

These functions can be either stationary or non-stationary. Stationary kernels depend on the distance between the two configurations being compared, meaning they are functions of $\mathbf{x}_i - \mathbf{x}_j$. When invoking a stationary covariance function, it is important that the data are also approximately stationary. That is, it must hold that the rate of change in the output with the inputs is constant (or at least approximately so) across the input space. The inputs or outputs are often transformed prior to training for this reason. For example, an $r \to r^{-1}$ transform on the interatomic distances is effective when modelling potential energy surfaces [27–29, 94].

A common example of a stationary kernel is the squared exponential function,

$$k_{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\mathrm{n}}^2 + \sigma_{\mathrm{f}}^2 \prod_{d=1}^{N_{\mathrm{d}}} \exp\Big(- \frac{(x_i^{(d)} - x_j^{(d)})^2}{2 l_d^2} \Big). \tag{2.15}$$

Here, N_{d} is the length of $\mathbf{x}_i$ and $\mathbf{x}_j$, while d runs over their elements. The hyperparameters of $k_{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j)$ are the signal variance σ_{f}^2 and lengthscales l_d, as well as the aforementioned Gaussian noise variance σ_{n}^2. The lengthscales are denoted l_d rather than l as each x_i x_j pair will have its own lengthscale. σ_{f}^2 tunes the amplitude of the GP predictions, which means that a small σ_{f}^2 gives rise to a model for which the largest and smallest output values are similar. Meanwhile, l_d tunes the degree of covariance between points in input space: a large l_d means that points that are far away vary strongly with each other, while a small l_d implies that only proximal points exhibit significant covariance. The third hyperparameter, σ_{n}^2, accounts for the noise in the training data and stabilises the inversion of $\overline{\mathbf{K}}$ by adding σ_{n}^2 along its diagonal.

These effects are illustrated when x is a scalar in figure 2.1, which was made at http://www.infinitecuriosity.org/vizgp/. The plots show that when the lengthscale is increased, the GP varies far less rapidly with changes in the value of the input. Moreover, increasing the signal variance leads to more variability in the y-values that fall within one or two standard deviations of the mean. Furthermore, as $\sigma_{\mathrm{n}}^2 \neq 0$ in this figure, the random sample from the GP does not pass exactly through the observed data.

Inclusive of noise, the log-likelihood becomes

$$\log(\mathcal{L}) = -\frac{N_{\mathrm{t}}}{2} \log(2\pi) - \frac{1}{2} \log |\overline{\mathbf{K}} + \bar{\mathbf{I}}\sigma_{\mathrm{n}}^2| - \frac{1}{2} \mathbf{Y}^{\mathrm{T}} (\overline{\mathbf{K}} + \bar{\mathbf{I}}\sigma_{\mathrm{n}}^2)^{-1} \mathbf{Y}. \tag{2.16}$$

While $-\frac{N_{\mathrm{t}}}{2}\log(2\pi)$ is a normalisation constant, the other two terms in equation (2.16) are interpretable in terms of the model hyperparameters. $-\frac{1}{2}\mathbf{Y}^{\mathrm{T}}(\overline{\mathbf{K}} +$

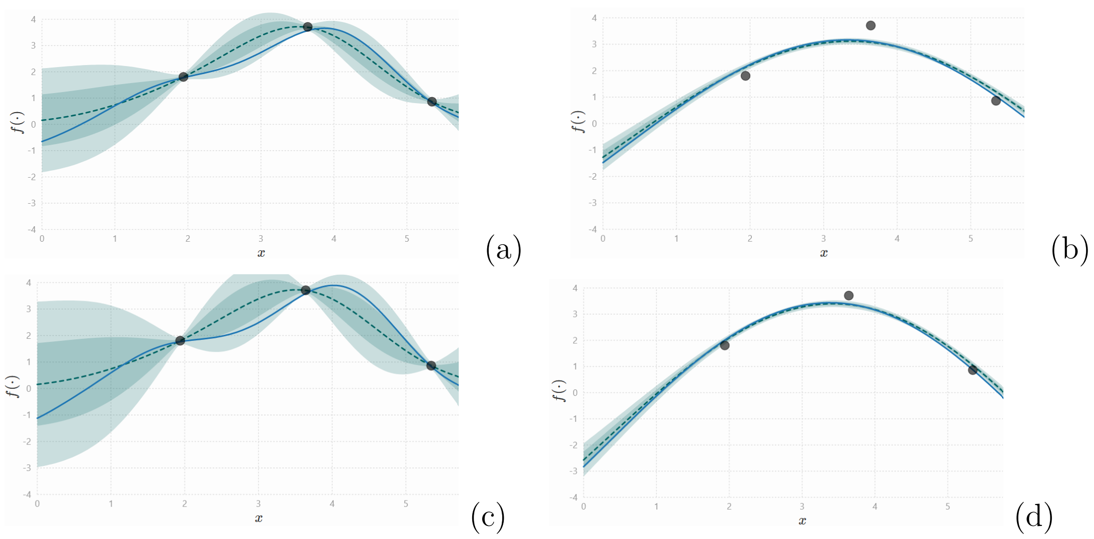


Figure 2.1: Plots showing the mean (dotted line) and a random sample (solid line) from a GP trained on three data points with $\sigma_{\mathrm{f}}^2 = l = 1$ (a), $\sigma_{\mathrm{f}}^2 = 1$ and $l = 4.5$ (b), $\sigma_{\mathrm{f}}^2 = 2.5$ and $l = 1$ (c), and $\sigma_{\mathrm{f}}^2 = 2.5$ and $l = 4.5$ (d). The lighter blue shaded region shows two standard deviations and the darker region a single standard deviation from the mean. All models assume $\sigma_{\mathrm{n}}^2 = 0.07$ and use a squared exponential kernel function.

$\overline{\mathbf{I}}\sigma_{\mathrm{n}}^2)^{-1}\mathbf{Y}$ is a data fit term and $-\frac{1}{2}\log|\overline{\mathbf{K}} + \overline{\mathbf{I}}\sigma_{\mathrm{n}}^2|$ penalises model complexity. As l increases, the complexity term becomes more positive because a larger lengthscale produces a less complex model. Meanwhile, increasing l makes the data-fit term more negative as the flexibility of the model is reduced. Balancing these terms means the model with the largest lengthscale that gives a good fit to the data is selected. Concurrently, models with minimal σ_{n}^2 are preferred as they reduce the magnitudes of $-\frac{1}{2}\mathbf{Y}^{\mathrm{T}}(\overline{\mathbf{K}} + \overline{\mathbf{I}}\sigma_{\mathrm{n}}^2)^{-1}\mathbf{Y}$ and $-\frac{1}{2}\log|\overline{\mathbf{K}} + \overline{\mathbf{I}}\sigma_{\mathrm{n}}^2|$. Thus GP models that are optimised using $\log(\mathcal{L})$ avoid overfitting: favouring large lengthscales avoids the development of models that pass through all training points at the expense of accuracy on unseen data, while penalising large Gaussian noise means that models that pass as close as possible to the observations are still preferred. This is a key advantage that GPs have over comparable methods such as neural networks (NNs).

Alternative kernel functions

Many alternatives to $k_{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j)$ exist. Some are discussed here with $N_{\mathrm{d}} = 1$ for clarity. A popular example of an alternative covariance function is the Matern kernel

$$k_{\mathrm{M}}(x_i, x_j) = \sigma_{\mathrm{f}}^2 \frac{2^{\nu-1}}{\Gamma(\nu)}\Big(\sqrt{2\nu}\frac{|x_i - x_j|}{p}\Big)^{\nu} K_{\nu}\Big(\sqrt{2\nu}\frac{|x_i - x_j|}{p}\Big) + \sigma_{\mathrm{n}}^2. \tag{2.17}$$

Here σ_{f}^2 and σ_{n}^2 are as before, ν is a positive integer, p is a hyperparameter akin to the lengthscale in $k_{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j)$, $\Gamma(\nu)$ is the gamma function

$$\Gamma(\nu) = (\nu - 1)!, \tag{2.18}$$

and K_{ν} is the modified Bessel function [95].

Unlike a GP that employs a squared exponential kernel function, which is infinitely differentiable, one that employs a Matern kernel is differentiable $\nu - 1$ times. However, as $\nu \to \infty$ the Matern kernel approaches the squared exponential kernel. Special cases of the Matern kernel appear when $\nu = n/2$, where n is a positive integer [95]. For example, when $\nu = 5/2$,

$$k_{\mathrm{M5/2}}(x_i, x_j) = \sigma_{\mathrm{f}}^2\Big(1 + \frac{\sqrt{5}|x_i - x_j|}{p} + \frac{25(x_i - x_j)^2}{3p^2}\Big)\exp\Big(-\frac{\sqrt{5}|x_i - x_j|}{p}\Big) + \sigma_{\mathrm{n}}^2. \tag{2.19}$$

In addition to stationary kernels, non-stationary kernels exist. These do not depend on the distance between the two configurations being examined, meaning they do not rely on the data being stationary. The multi-layer perceptron kernel, or neural network kernel, is an example of such a covariance function. It has the form

$$k_{\mathrm{NN}}(x_i, x_j) = \sigma_{\mathrm{f}}^2 \frac{2}{\pi} \arcsin\Big(\frac{\sigma_{\mathrm{w}}^2 x_i x_j + \sigma_{\mathrm{b}}^2}{\sqrt{\sigma_{\mathrm{w}}^2 x_i^2 + \sigma_{\mathrm{b}}^2 + 1}\sqrt{\sigma_{\mathrm{w}}^2 x_j^2 + \sigma_{\mathrm{b}}^2 + 1}}\Big) + \sigma_{\mathrm{n}}^2, \tag{2.20}$$

where σ_{b}^2 and σ_{w}^2 are additional hyperparameters.

All kernel functions must be positive semi-definite. It follows that combining two or more kernel functions together through addition and multiplication will result in another acceptable kernel function. Consequently, the functions introduced above can be thought of as building blocks to create new kernels tailored to the problem being modelled. This makes GPs flexible modelling tools, though a convoluted kernel function will possess many hyperparameters, which will result in lengthy re-optimisations.

Such an approach has been applied to the development of GP potentials previously, with the resultant composite kernels [35] optimised based on their Bayesian information criterion (BIC) [96] rather than the log-likelihood,

$$\mathrm{BIC} = \log(\mathcal{L}) - \frac{M}{2}\log(N_{\mathrm{t}}). \tag{2.21}$$

Because M is the number of hyperparameters in the kernel function, the BIC penalises convoluted kernels.

2.1.4 Gaussian process potentials for intermolecular interactions

Gaussian process potentials have been developed for various systems, from intramolecular potentials of water [55] and various amino acids [56, 57] to intermolecular potentials for many solid systems [40–42, 60–63]. These potentials have been successfully employed in a range of applications, such as the simulation of amorphous SiH [97] and the estimation of non-negligible three-body effects on the intramolecular potential of water [58].

This work is concerned with developing intermolecular potentials for application to systems in the gas or liquid phases. It therefore expands the work of Uteva *et al.* [27, 29], an overview of which is given in this section. The methods discussed have been used to produce successfully GP potentials for various systems by training on a Latin hypercube [27] or using sequential design [29].

Regardless of training set design, a symmetric version of the squared exponential covariance function [27, 29]

$$k_{\mathrm{Sym}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\mathrm{p}\in\mathrm{P}} k_{\mathrm{SE}}(\mathbf{x}_i, \mathrm{p}\mathbf{x}_j), \tag{2.22}$$

was invoked. Here, P is the set of all permutations of the interatomic distances under which the energy is invariant and p is a single permutation within this set. $k_{\mathrm{Sym}}(\mathbf{x}_i, \mathbf{x}_j)$ therefore exploits the symmetric nature of potential energy surfaces by ensuring equivalent distances can be swapped with no variation in the energy prediction. For example, the CO_2-Ne interaction is described fully by the three

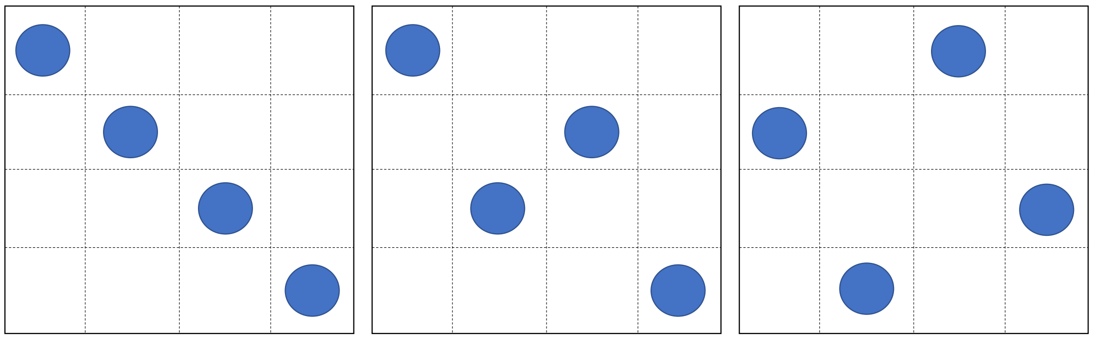


Figure 2.2: Diagram showing three possible Latin square samples of four points from a two-dimensional input space. This is a copy of a figure from the thesis of Elena Uteva [98].

distances $R_{\mathrm{O}_1-\mathrm{Ne}}$, $R_{\mathrm{C}-\mathrm{Ne}}$ and $R_{\mathrm{O}_2-\mathrm{Ne}}$. As the first and last of these distances are equivalent, there is no change in energy when they are interchanged. That $k_{\mathrm{Sym}}(\mathbf{x}_i, \mathbf{x}_j)$ accounts for this permits training of a GP potential that can predict any point on a PES from training data that cover only the symmetry-distinct subspace of that PES. For CO_2-Ne this subspace comprises the region where $R_{\mathrm{O}_1-\mathrm{Ne}} < R_{\mathrm{O}_2-\mathrm{Ne}}$.

Equation (2.22) conveys that $k_{\mathrm{Sym}}(\mathbf{x}_i, \mathbf{x}_j)$ is a stationary kernel. Thus, the aforementioned $r \to r^{-1}$ transform was applied to all distances used in training. That is, $\mathbf{x}_i$ and $\mathbf{x}_j$ were vectors of inverse interatomic distances. It was found that such a transform approximated stationarity in the training data to an extent sufficient to improve vastly the efficiency of training when used in conjunction with $k_{\mathrm{Sym}}(\mathbf{x}_i, \mathbf{x}_j)$ [27]. When r is low r^{-1} varies quickly, meaning both r^{-1} and the energy E vary rapidly in the repulsive wall; for large r, meanwhile, r^{-1} has a far smaller rate of change that matches the gradual change in E at large separations. Thus the rates of change of the input (r^{-1}) and output (E) are similar across the PES under this transform. This is in contrast to non-inverse distances, for which the rate of change in E with r varies dramatically between small and large separations.

Latin hypercube training

A Latin hypercube (LHC) [64–66] is a space-filling design that samples input space more evenly than random sampling. In two dimensions a LHC becomes Latin square, which places points such that there is a sample in each row and column, as illustrated in figure 2.2.

The training set $\mathcal{T} = \{\mathbf{x}_i, \mathrm{Y}_i\}_{i=1}^{N_\mathrm{t}}$ was built using a LHC that covered the symmetry-distinct region of input space. This was done by setting the parameters that define the LHC to specify the symmetry-distinct region. For example, when training on the CO_2-Ne potential, the maximum value for the angle between the axis of the CO_2 and the Ne was $\frac{\pi}{2}$ radians. This ensures that the Ne is always closer to one of the O atoms.

Minimum and maximum interatomic distances, R_min and R_max respectively, were also imposed upon the training LHCs. As the LHCs were specified in inverse distances, no sample point could be placed if all inverse distance therein were less than R_max^{-1} or any exceeded R_min^{-1}. In all cases $R_\mathrm{min} = 1.5$ Å, while $R_\mathrm{max} = 8.5$ Å. In addition, a high-energy cut-off E_cut of 5 x 10^{-3} Hartrees (E_h) was applied to LHCs for all potentials apart from that of $(HF)_2$, which had $E_\mathrm{cut} = 0.02\ E_\mathrm{h}$. E_cut was applied after the LHC was built in all cases and was chosen relative to the well-depth of the interaction for which the LHC was built (*e.g.* $(HF)_2$ had a larger well-depth than the other systems so its E_cut was larger).

Even in the relatively straightforward case given in figure 2.2, however, there are many possible ways to sample the input space in a LHC design. To circumvent this issue, thousands of LHCs were produced for each system, referred to as candidate LHCs. Of these candidate LHCs, the one with the largest minimum separation was chosen as $\mathcal{T}$. This 'maximin' LHC was considered to sample the input space more evenly than the other candidates, with the rightmost example in figure 2.2 depicting the 'maximin' Latin square. Energy calculations were only undertaken on the 'maximin' LHC, with the distance between the configurations $\mathbf{x}_i$ and $\mathbf{x}_j$ defined as

$$|r|_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^\mathrm{T}(\mathbf{x}_i - \mathbf{x}_j). \tag{2.23}$$

It was shown that training a GP potential on LHC data developed accurate potentials for the CO_2-Ne, CO_2-H_2, CO_2-CO, $(HF)_2$ and CH_4-N_2 systems. Moreover, the work demonstrated that the use of the symmetric kernel and $r \to r^{-1}$ transform on the inputs improved the quality of the potentials at a given N_t. In addition, it was found that the GP potential for the CO_2-CO potential energy surface could be employed to estimate the CO_2-CO second virial coefficient with good agreement with experimental observations. This last result was achieved by upgrading the LHC data for the CO_2-CO systems from second order Møller-Plesset perturbation theory (MP2) [99] to coupled cluster singles, doubles and perturbative triples (CCSD(T)) [100–103]. Both of these methods are discussed further in Section 2.2, but this information regardless demonstrates a successful use of GP potentials with transfer learning [51, 52].

The accuracy of the aforementioned virial calculation was improved markedly by introducing a long-range function for prediction at any configurations with separations exceeding R_{max}. Thus this distance represented a fixed cross-over distance R_{cross} between the GP and this function. In other words, $R_{\text{cross}} = R_{\text{max}}$ in the work of Uteva *et al.*. The long-range function was introduced because the GP tends to a non-zero constant at large separations [27], despite the mean function to which it should tend being zero everywhere. This is likely a consequence of the $r \to r^{-1}$ transform. A long-range point that is far from the training data in r will still be close in r^{-1}, meaning the GP will not make a prediction with the mean function but as an approximate average of the training energies. The average is taken because all training points will be at similar separations from the long-range point, meaning their covariance with this point will be similar. This average is often positive due to the larger magnitude of the positive energies in the repulsive wall.

Sequential design

In a later work, Uteva *et al.* [29] presented active learning [104, 105] and sequential design [67, 68] approaches to training, in which the training set was built iteratively. In this approach, training and validation of the GP models involved three data sets: a training set $\mathcal{T}$, a reference set $\mathcal{R}$ and a test set $\mathcal{S}$. The training set was used in GP regression and re-optimisation; the reference set was a LHC that provided a pool of configurations from which new training points were selected; and the test set was a grid or LHC used to determine the accuracy of the potential against independent data. $\mathcal{R}$ was built to the same specifications outlined in the overview of LHC training, as was $\mathcal{S}$. All LHCs were specified in the same way as those in the previous section, meaning it was once again fixed that $R_{\text{max}} = R_{\text{cross}} = 8.5$ Å.

Sequential design and active learning methods select training points iteratively (or sequentially) from $\mathcal{R}$ to fill $\mathcal{T}$ before re-optimising the GP hyperparameters. That process, referred to here as a stage of training, was repeated until the model achieved the desired predictive accuracy on $\mathcal{S}$. There are various criteria for selecting the training points, but regardless training by sequential design or active learning prevents the placement of training data in regions where the GP already has high predictive accuracy. This is beneficial as such points do little to improve accuracy but still increase the size of $\mathcal{T}$. It was found that for a given N_t, the error on $\mathcal{S}$ achieved by a potential from one of these methods was lower than that attained by LHC training [29].

The difference between active learning and sequential design is that the former does not require outputs to be calculated until reference points are added to $\mathcal{T}$. Thus, active learning can be envisioned as a subset of sequential design. When large amounts of computationally intensive reference data are required, active learning is an attractive prospect. Examples of active learning methods for training GPs include those of MacKay [104] and Cohn [105]. The first of these added the configuration where the variance of the GP was highest in $\mathcal{R}$ to $\mathcal{T}$ (*i.e.* a highest variance search). Thus, it exploited that GPs have a closed form for the variance, as shown earlier in equation (2.14). This property was also utilised by Cohn, who proposed choosing the configuration that would reduce the average variance over the test set as the training point at each stage. Moreover, Uteva *et al*. proposed the 'two set search' method [29]. This entailed training two GPs from a different initial $\mathcal{T}$ and adding the point in $\mathcal{R}$ where the two had the largest discrepancy in their predictions to the $\mathcal{T}$ of each. More widely, active learning strategies have been used to develop GP [29, 40, 106], moment tensor [25] and neural network [23] potentials successfully.

However, calculating energies at thousands of configurations for the small intermolecular systems explored by Uteva *et al* at the MP2 level of theory is inexpensive computationally compared to training the GP potential. Thus the advantage of active learning over sequential design in this context is negligible. In addition, GP potentials were trained most efficiently via either the two set search active learning strategy or a highest error search sequential design method [29].

The latter entailed adding to $\mathcal{T}$ the configuration in $\mathcal{R}$ where the squared error in the GP prediction was highest. As the two set search relies on re-optimising two GPs at each stage of training, the time taken to use this method would exceed that of the highest error search even accounting for the time for the MP2 calculations on the reference set. Moreover, the highest error search was found to match the two set method in terms of predictive accuracy for a given N_{t} for the CO_2-Ne and CO_2-H_2 systems, and outperform it for training on the Ar_3 PES.

As mentioned, the utility of GP potentials in applications is dependent upon their training sets being as small as possible to increase the speed of prediction. Furthermore, smaller training sets are more easily upgraded to more accurate levels of theory. This means that in the development of the most accurate GP potentials, which are among the most accurate machine-learned potentials, it is always beneficial to reduce the size of the training set.

2.2 Potential energy surfaces

Having discussed GP potentials, it is instructive to next introduce the context in which they will be applied here. Potential energy surfaces (PESs) are a fundamental concept in chemistry and describe how the energy of a system of electrons and fixed nuclei varies with configuration. That PESs are crucial is reflected in the breadth of their applications, with recent examples including simulating N_2-N and N_2-N_2 interactions [107], and computing rate coefficients for the oxidation of diethyl ether [108].

The importance of PESs is further reflected in the variety of methods that have been developed to approximate them. The most accurate predictions of the energy at a given point on a PES come from electronic structure calculations such as Møller-Plesset perturbation theory [99] or coupled cluster theory [100–103], which are both based on Hartree-Fock theory [109]. As these methods require no experimental data they are often referred to as *ab initio* techniques, although Hartree-Fock theory does require an empirical correction for dispersion interactions. An alternative method is density functional theory [110], which though largely predicated on first principles requires an empirical approximation of dispersion interactions. This means it is not truly *ab initio*, though it is still referred to as such here.

This section outlines where the concept of PESs comes from, how *ab initio* methods are used to approximate energies on a PES, the theory underpinning such methods, and why the scaling of the computational cost of prediction with these techniques is so poor. In addition to the references already given in this chapter, the subsequent discussion is based on Chapters 3-6 of 'Introduction to Computational Chemistry' [111] and Chapter 2 of the PhD thesis of Elena Uteva [98].

2.2.1 Origins of potential energy surfaces

The energy of any system of electrons and nuclei is governed by the interactions between these particles. Thus, when estimating the energy of such a system, it is paramount that their behaviour is well-described. This behaviour is encapsulated by a wavefunction, Ψ, which is a function of the electronic and nuclear coordinates, $\mathbf{r}$ and $\mathbf{R}$ respectively.

The Born-Oppenheimer approximation [112] postulates that the electrons are vastly lighter and faster than the nuclei. Thus, the former can respond instantaneously to any variation in nuclear position. This allows Ψ to be partitioned into electronic and nuclear parts,

$$\Psi(\mathbf{r},\mathbf{R}) = \Psi_{\mathrm{el}}(\mathbf{r};\mathbf{R})\Psi_{\mathrm{nu}}(\mathbf{R}), \tag{2.24}$$

with the nuclear coordinates $\mathbf{R}$ parameters in Ψ_{el}.

Under the Born-Oppenheimer approximation the energy of the system can be found for a given set of nuclear coordinates via the time-independent electronic Schrödinger equation,

$$\hat{H}_{\mathrm{el}}\Psi_{\mathrm{el}}(\mathbf{r};\mathbf{R}) = E\Psi_{\mathrm{el}}(\mathbf{r};\mathbf{R}). \tag{2.25}$$

Here, $\hat{H}_{\mathrm{el}}$ is the electronic Hamiltonian operator, which accounts for the electron kinetic energies, the electron-electron potential energy, the electron-nuclei potential energy and the internuclear potential energy. As the nuclear positions are fixed, the nuclear kinetic energies are omitted.

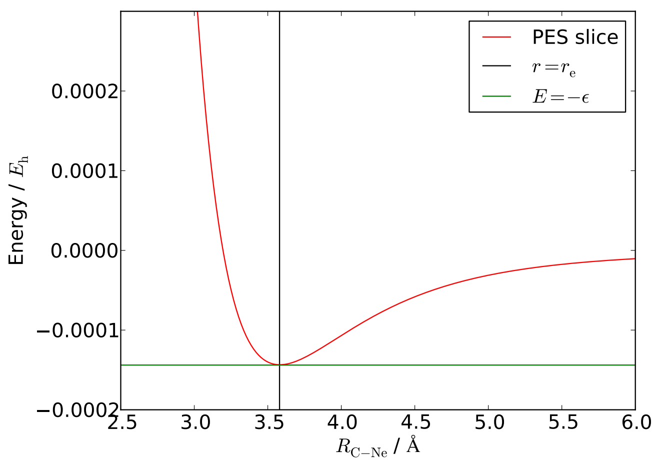


Figure 2.3: Plot showing a slice through the PES of the CO-Ne interaction, with the C-O distance fixed and the particles arranged such that $R_{\mathrm{C-Ne}} = R_{\mathrm{O-Ne}}$. Also shown are some key features of a PES, including a line showing the well-depth ϵ (green) and the equilibrium separation in this conformation r_e (black). The PES slice itself comes from predictions by a GP trained on MP2 data.

Consequently, under the Born-Oppenheimer approximation, a PES that shows the variation of the energy with nuclear geometry can be constructed, with a slice through such a surface shown in figure 2.3. Though this approximation introduces errors, these are negligible for most systems [111]. Systems for which the Born-Oppenheimer approximation is inadequate are those in which two electronic states are separated by an energy that is less than the nuclear kinetic energy.

2.2.2 Estimating the form of the wavefunction

Exact solutions to the Schrödinger equation cannot be found for systems of more than one electron. Estimates of the solution for more complex systems are obtained by first approximating the wavefunction. Any approximate wavefunction must obey the Pauli principle, which states that two electrons cannot be defined by an identical set of quantum numbers. Consequently the sign of the wavefunction must alter under the interchange of any two electrons.

For a two-electron system, the wavefunction that obeys the Pauli principle can be written as

$$\Psi_{\mathrm{el}}(r_1, r_2; \mathbf{R}) = \frac{1}{\sqrt{2}}[\psi_1(r_1)\psi_2(r_2) - \psi_1(r_2)\psi_2(r_1)], \tag{2.26}$$

where $\psi_i(r_j)$ denotes the spin orbital of the electron at position r_j. It can be verified that this wavefunction obeys the Pauli principle by switching electrons one and two,

$$\begin{aligned}\Psi_{\mathrm{el}}(r_2, r_1; \mathbf{R}) &= \frac{1}{\sqrt{2}}[\psi_1(r_2)\psi_2(r_1) - \psi_1(r_1)\psi_2(r_2)] \\ &= -\Psi_{\mathrm{el}}(r_1, r_2; \mathbf{R}).\end{aligned} \tag{2.27}$$

For a system of N_e electrons, the spin orbitals are held in a Slater determinant,

$$\Psi_{\mathrm{el}} = \frac{1}{\sqrt{N_\mathrm{e}!}} \begin{vmatrix} \psi_1(r_1) & \psi_2(r_1) & \cdots & \psi_{N_\mathrm{e}}(r_1) \\ \psi_1(r_2) & \psi_2(r_2) & \cdots & \psi_{N_\mathrm{e}}(r_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(r_{N_\mathrm{e}}) & \psi_2(r_{N_\mathrm{e}}) & \cdots & \psi_{N_\mathrm{e}}(r_{N_\mathrm{e}}) \end{vmatrix}. \tag{2.28}$$

When considering a molecule, each column in the determinant represents a molecular orbital ϕ_i. The rows, meanwhile, show the electronic positions.

The molecular orbitals are defined conventionally as a linear combination of atomic orbitals (LCAO),

$$\phi_i = \sum_{\alpha}^{N_{\mathrm{basis}}} c_{\alpha i}\psi_\alpha, \tag{2.29}$$

where $c_{\alpha i}$ is a molecular orbital coefficient and ψ_α is an atomic orbital. Basis functions are used to estimate the form of ψ_α, with N_{basis} the number of basis functions employed. A basis function must simultaneously reflect the physics of the system being investigated and be computationally tractable. The former suggests the use of functions that are exact solutions to the Schrödinger equation for an electron around a nucleus, however these are intensive computationally.

Instead, Gaussian-type orbitals (GTOs) are used widely as basis functions in electronic structure calculations,

$$\chi_{\mathrm{GTO}}(r, \theta, \phi) = N Y_{\mathrm{l,m}}(\theta, \phi) r^l \mathrm{e}^{-\zeta r^2}. \tag{2.30}$$

In equation (2.30), N is a normalising constant, $Y_{\mathrm{l,m}}(\theta, \phi)$ is a spherical harmonic function that defines the type of orbital the electron occupies, l is the angular quantum number, r is electron-nuclear separation and ζ can be varied to optimise the orbital.

GTOs are not solutions to the Schrödinger equation for an electron around a nucleus, hence a single GTO is too smooth and flat at the nucleus and decays too quickly with electron-nuclear separation to capture accurately the behaviour of an electron in this environment. However, a linear combination of GTOs can approximate this behaviour sufficiently. Though a larger number of basis functions leads to more expensive calculations, GTOs are so computationally efficient that a collection of GTOs will facilitate faster calculations than a single basis function obtained from a solution to the Schrödinger equation.

The collection of basis functions used in an *ab initio* calculation is referred to as the basis set. The computational cost of any such calculation scales as $\mathcal{O}(N_{\text{basis}}^4)$. Though this can be reduced to the order of N_{basis}^3 or for large systems N_{basis}^2, it is paramount that the smallest possible basis set is chosen. At a minimum, N_{basis} must equal the number of electrons in the system. To increase accuracy, however, the number of basis functions used for each valence electron can be doubled, tripled, *etc.*. The first case is known as a double zeta basis set in which the valence electrons are described using two basis functions, each with different directionality. Each of these basis functions will comprise multiple (*e.g.* 3-6) GTOs.

A commonly used collection of basis sets are the correlation-consistent basis sets of Dunning *et al.* [113]. These are generally denoted as cc-pVNZ, where 'cc-p' stands for correlation-consistent polarised, 'V' shows a split-valence set in which further basis functions are added to valence electrons, and 'N' is the number of basis functions on those electrons. Such basis sets are referred to as 'correlation-consistent' as they account for the effects of one valence electron on the movement of another (*i.e.* their correlation). These basis sets may also be prefixed with the term 'aug', which indicates the presence of extra diffuse basis functions. These functions better represent the areas of the molecular orbitals that are further away from the nucleus. Meanwhile, a cc-pCVNZ basis set includes all correlation between core electrons and core electrons with valence electrons, and a cc-pwCVNZ basis set includes all valence-core correlation and a small amount of core-core correlation [114]. As intermolecular PESs, which are the focus of this work, are low in energy, they often require large basis sets such as aug-ccpVTZ.

Regardless of which is employed in a calculation, the basis set is the estimate of the form of the wavefunction. Due to the aforementioned poor scaling with N_{basis}, estimating the energy of a system with many electrons and highly occupied valence orbitals will incur a large computational cost.

2.2.3 Electronic structure theories

A basis set must be parameterised once it has been chosen to approximate the wavefunction. This parameterisation is undertaken via an electronic structure, or *ab initio*, theory. The simplest of these methods is Hartree-Fock theory, which relies on the variational principle. This states that the energy of an approximate wavefunction will always exceed that of the real function. Thus, by minimising the energy of the system one concomitantly optimises the parameters of the basis set. The energy of a system with a given Hamiltonian and approximate wavefunction is given by

$$E = \frac{\langle \Psi_{\text{el}} | \hat{H} | \Psi_{\text{el}} \rangle}{\langle \Psi_{\text{el}} | \Psi_{\text{el}} \rangle}, \tag{2.31}$$

using standard Dirac notation. Given that the approximate wavefunctions discussed earlier are normalised, $E = \langle \Psi_{\text{el}} | \hat{H} | \Psi_{\text{el}} \rangle$. Henceforth, the electronic wavefunction will be denoted Ψ_{EST} for brevity, where EST denotes an electronic structure theory. The notation depicting the dependence on electronic and nuclear positions of Ψ_{el} is also dropped for this reason.

The theories introduced here are discussed in order of increasing accuracy. In all cases, however, their computational expense prohibits their use in simulations. The cheapest scale with N_{basis}^4, which can be somewhat circumvented by calculating the energies of small clusters of particles rather than a whole simulation box. However, the expense is still massive and the most accurate methods scale as $\mathcal{O}(N_{\text{basis}}^8)$, which leaves no workable method for offsetting the cost.

Hartree-Fock theory

Hartree-Fock theory (HF) is the simplest electronic structure theory and forms the basis for many of the others that will be discussed subsequently. This method assumes that the electrons move independently of one another, experiencing only a mean field of all other electrons. Therefore in the HF Hamiltonian, $\hat{H}_{\text{HF}}$, the electron-electron potential operator is a mean field of all electrons. That is, HF theory does not account for electron correlation. Electron correlation is a stabilising effect as it reduces the probability of two electrons being proximal to each other, which induces repulsion. As such, ignoring electron correlation leads to the calculated energy being an overestimate.

The use of the variational principle allows HF calculations to be solved self-consistently, which has led to this method being described as a self-consistent field (SCF) method. SCF methods begin with an initial guess of the parameters of Ψ_{HF} that is updated iteratively until the energy converges, giving the final HF wavefunction. The accuracy of the converged energy is dependent on the size of the basis set. This size also dominates the cost of calculation, which scales as $\mathcal{O}(N_{\text{basis}}^4)$.

Møller-Plesset perturbation theory

Møller-Plesset perturbation theory (MP) postulates that accounting for the electron correlation requires only a marginal amendment to the method of determining the energy of the system without this effect. Thus, the electron correlation energy can be estimated as a small change (or perturbation) to the energy from a HF calculation.

Under this assumption, the perturbed Schrödinger equation is written

$$\hat{H}_{\text{MP}}\Psi_{\text{MP}} = E_{\text{MP}}\Psi_{\text{MP}}, \tag{2.32}$$

where

$$\hat{H}_{\text{MP}} = \hat{H}_{\text{HF}} + \lambda\hat{H}'_{\text{HF}}, \tag{2.33}$$

is the MP Hamiltonian. The perturbation $\hat{H}'_{\text{HF}}$,

$$\hat{H}'_{\text{HF}} = V_{\text{ee}} - 2\langle V_{\text{ee}}\rangle, \tag{2.34}$$

is the fluctuation potential. This takes the difference between the true electron-electron repulsion V_{ee} and twice the average repulsion predicted in HF $\langle V_{\text{ee}}\rangle$. That is, the MP Hamiltonian is equivalent to the HF Hamiltonian with a correction to account for electron correlation. Meanwhile,

$$E_{\text{MP}} = E_{\text{HF}} + \sum_{i=1}^{n} \lambda^n E_{\text{HF}}^{(n)} \tag{2.35}$$

is the perturbed energy and

$$\Psi_{\text{MP}} = \Psi_{\text{HF}} + \sum_{i=1}^{n} \lambda^n \Psi_{\text{HF}}^{(n)} \tag{2.36}$$

is the perturbed wavefunction. In these equations λ is a parameter controlling the size of the perturbation, and n is the total number of terms in the power series in λ used for the corrections to the wavefunction and Hamiltonian. For this reason MP is generally denoted as MPn, meaning a calculation featuring a second-order correction is referred to as an MP2 calculation. The additional calculation of the perturbations means that MP2 calculations scale as $\mathcal{O}(N_{\text{basis}}^5)$.

Coupled cluster theory

Coupled cluster theory (CC) accounts for electron correlation through inclusion of the effects of exciting electrons from their ground state orbitals. This uses the operator $\hat{T}$, which when truncated to double excitations is

$$\hat{T} = \hat{T}_1 + \hat{T}_2. \tag{2.37}$$

Here $\hat{T}_1$ covers all single excitations (*i.e.* excitations of one electron) and $\hat{T}_2$ all double excitations. The resultant coupled cluster wavefunction is

$$\Psi_{\text{CC}} = \exp(\hat{T})\Psi_{\text{HF}}. \tag{2.38}$$

In equation (2.38), $\exp(\hat{T})$ denotes the series expansion of $\hat{T}$,

$$\exp(\hat{T}) = 1 + \hat{T} + \frac{1}{2}\hat{T}^2 + ... \tag{2.39}$$

This implies that, for a coupled cluster calculation that includes only single and double excitations (*i.e.* a coupled cluster singles-doubles, or CCSD, calculation), the operators $\hat{T}_1$ and $\hat{T}_2$ are included up to infinite powers. However, in practice excitations of order four or higher do not alter the energy. Therefore terms such as $\hat{T}_1^4$, which indicates excitations of four non-interacting electrons, are not included.

Despite this, the cost of a CCSD calculation scales as $\mathcal{O}(N_{\text{basis}}^6)$, though it gives a more accurate approximation of the energy than the aforementioned *ab initio* methods. Adding triple excitations leads to a scaling of order $\mathcal{O}(N_{\text{basis}}^8)$, which is prohibitively expensive for all but the smallest systems. Triple excitations can be assumed to be a perturbation to the CCSD energy, just as MP assumes electron correlation is a perturbation to E_{HF}, and calculated using CCSD(T). The brackets denote that the effect of the triple excitations is treated as a perturbation.

Regardless, the high cost of CCSD(T) calculations, which scale as $\mathcal{O}(N_{\text{basis}}^7)$, means that it is desirable to undertake as few as possible. This melds well with the capacity of GPs to model PESs with few training points relative to other methods. A minimal training set can be developed via sequential design with reference MP2 calculations, which are then 'upgraded' to CCSD(T) energies to give a more accurate potential. This is the basis of the transfer learning approach discussed in Section 2.1.4.

2.2.4 Density functional theory

The aforementioned electronic structure theories were all reliant on estimating the wavefunction that describes the electronic behaviour of a system. Density functional theory (DFT), meanwhile, approximates the energy of a system as a functional of the electron density. Whereas a function returns a single value from a set of variables, a functional returns a single value from a function. Hence the DFT energy is denoted $E_{\text{DFT}}[\rho(\mathbf{r})]$, where $\rho(\mathbf{r})$ is the electron density.

Dropping the **r**-dependence of ρ for brevity, the DFT energy is calculated as

$$E_{\text{DFT}}[\rho] = T_{\text{S}}[\rho] + E_{\text{en}}[\rho] + E_{\text{nn}}[\rho] + J[\rho] + E_{\text{xc}}[\rho]. \tag{2.40}$$

The first four terms in equation (2.40) are the electron kinetic energy, the electron-nuclear potential, the electron-electron potential and the internuclear potential. Forms for all of these terms are known in Kohn-Sham DFT [115], which accounts for the electron kinetic energies via the introduction of non-interacting orbitals, known as Kohn-Sham orbitals. The subscript "S" in the kinetic energy term denotes that this quantity is calculated from a Slater determinant.

The final term, referred to as the exchange-correlation energy, is unknown and accounts for the effect of the correlation between electrons. $T_{\text{S}}[\rho]$ and $J[\rho]$ correspond to the kinetic energy and electron-electron interaction energy without correlation. Thus one can write

$$E_{\text{xc}}[\rho] = (T[\rho] - T_{\text{S}}[\rho]) + (E_{\text{ee}}[\rho] - J[\rho]), \tag{2.41}$$

where $T[\rho]$ and $E_{\mathrm{ee}}[\rho]$ are the true (but unknown) electron kinetic energy and electron-electron repulsion energy.

Any DFT calculation requires an estimate of $E_{\mathrm{xc}}[\rho]$. Though this is often smaller than the other contributions, it is still significant in many systems, such as in intermolecular potentials. However, said estimate may be made from experimental observations or different *ab initio* data. This dependence means DFT is not a purely *ab initio* method, though it serves the same purpose as the aforementioned techniques and will still be referred to with this term here.

The advantage of DFT over MP and CC theories is its computational scaling, which is similar to that of HF theory. This has meant that DFT has become the technique of choice in *ab initio* molecular simulations. However, though it scales better than other *ab initio* methods, DFT-based simulations are still limited to a few hundred atoms. Furthermore, DFT accounts for correlation with an empirical term $E_{\mathrm{xc}}[\rho]$, which renders it a poor choice for finding the energy of intermolecular interactions where dispersion interactions are significant. Gaussian process potentials trained via transfer learning on data from coupled cluster theory therefore offer a more fast and accurate route to employing first principles predictions in simulations.

2.3 Intermolecular interactions

With the concept of potential energy surfaces introduced, an overview of the interactions that contribute to the energy at a given point thereupon is given in this section. These interactions between atoms can be divided into two general groups: covalent and non-covalent interactions. The former refers to the strong, attractive interactions between atoms that lead to bond formation, while the latter encompasses weaker, non-bonding interactions. Thus this chapter discusses the non-covalent interactions only as these inform the intermolecular potential energy surface between two or more molecules.

The section begins with a discussion of various types of intermolecular interactions, beginning with the strongest. These are presented in terms of additive interactions for simplicity. This is followed with overviews of the supermolecule approach, and additive and non-additive interactions. The section concludes with an overview of the multipole expansion, which is used to develop the long-range functions that take over prediction at large separations.

The goal is to explain intermolecular interactions such that their role in the potentials modelled with Gaussian processes in later chapters is clear. The discussion is based on Chapters 3 and 7 of Stone's 'The Theory of Intermolecular Forces' [116] and Chapters 3-8 of Israelachvili's 'Intermolecular and Surface Forces' [117].

2.3.1 Coulombic interactions

Coulombic interactions occur between two ions and are the strongest non-covalent interactions. Coulomb's law states that the force on one charged particle from the electric field of another at separation r is

$$F_{\mathrm{C}} = \frac{Q_1 Q_2}{4\pi\epsilon_0 r^2}, \tag{2.42}$$

where Q_i is the charge on particle i and ϵ_0 is the dielectric constant of the medium.

The work in later chapters is concerned with intermolecular energies, not forces. As the force is the negative of the derivative of the energy with respect to position, the relevant equation comes from integrating equation (2.42). This yields

$$E_{\mathrm{C}} = -\int_{\infty}^{r} \frac{Q_1 Q_2}{4\pi\epsilon_0 r^2} = \frac{Q_1 Q_2}{4\pi\epsilon_0 r}, \tag{2.43}$$

where the lower limit is ∞ because the energy is zero at infinite separation.

Equation (2.43) demonstrates that $E_{\mathrm{C}} \propto r^{-1}$. As such, the energies of Coulombic interactions decay slowly with the separation between the particles relative to the interactions discussed subsequently. It also illustrates that the energy of interaction between two oppositely charged ions will be attractive, while two ions of like charge will repel each other.

2.3.2 Charge- and dipole-dipole interactions

Charge-dipole and dipole-dipole intteractions all contain at least one polar molecule. While a polar molecule may possess no net charge and undergo no charge-charge interactions, a disparity in the distribution of electrons across its atoms may lead to the development of an electric dipole. A simple example of such a molecule is HF, on which the propensity of the fluorine to draw electron density onto itself leaves it with a partial negative charge and the hydrogen with a positive charge of equal magnitude.

The dipole moment μ of a polar molecule is given by $\mu = ql$, where l is the separation between the two charges $+q$ and $-q$. The strongest interaction a dipole moment will undergo is the charge-dipole interaction, for which the energy is

$$E_{\mathrm{CD}} = -\frac{\mu Q}{4\pi\epsilon_0 r^2}\cos\theta. \tag{2.44}$$

As the dipole is aligned along the polar molecule, an angular term cosθ is introduced to account for the orientation of the dipole relative to the ion.

Two dipoles, μ_1 and μ_2, can also interact with each other. The energy of a dipole-dipole interaction at separation r aligned relative to an axis is

$$E_{\mathrm{DD}} = -\frac{\mu_1\mu_2}{4\pi\epsilon_0 r^3}(2\cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2\cos\phi), \tag{2.45}$$

where θ_i is the in-plane angle between μ_i and the axis, and $\phi = \phi_1 - \phi_2$ is the difference between the out-of-plane angles of both dipoles. If $\theta_1 = \theta_2 = 0$ the dipoles are in line and have the most favourable interaction; if $\theta_1 = \theta_2 = \frac{\pi}{2}$ then the dipoles are parallel.

At standard temperatures, that $E_{\mathrm{DD}} \propto r^{-3}$ results in dipole-dipole interactions failing to influence markedly the alignment of two polar molecules for $r > 3.5$ Å [117]. Hydrogen bonds are, however, a special case of the dipole-dipole interaction for which this is not the case. They occur between polar molecules containing a hydrogen that is bonded to a strongly electronegative atom, such as fluorine or oxygen. The effects of hydrogen bonding are prevalent in water for this reason, as each oxygen is bonded to two hydrogen atoms. Their strength is a product of the small size of the hydrogen atom, which permits electronegative atoms on other molecules close access to the positively charged region of the dipole. Despite their significance in the behaviour of such an important molecule, the energy of hydrogen bonds is poorly understood and no simple equation has been found that predicts accurately their strengths. However, it is thought that $E_{\mathrm{HB}} \propto r^{-2}$, similarly to charge-dipole interactions.

In any molecule that possesses a permanent dipole, higher-order permanent multipoles will also be present. The next largest is the quadrupole, then the octopole and so on. The long-range functions used in Chapter 3 do not consider any multipoles higher than a quadrupole so they are not discussed here.

The energy of interaction of a quadrupole Θ with a dipole is

$$E_{\mathrm{DQ}} = -\frac{\mu_1 \Theta_2}{4\pi\epsilon_0 r^4}[1.5(\cos\theta_1(3\cos^2\theta_2 - 1) - \sin\theta_1 \sin\theta_2 \cos\phi)], \tag{2.46}$$

where the angles θ_i and ϕ are as before. Quadrupole-quadrupole interactions can also make significant contributions to the intermolecular potential. The energy of these interactions is

$$\begin{aligned} E_{\mathrm{QQ}} = &-\frac{\Theta_1\Theta_2}{4\pi\epsilon_0 r^5}[0.75(1 - 5\cos^2\theta_1 - 5\cos^2\theta_2 - 15\cos^2\theta_1\cos^2\theta_2 + \\ &2(4\cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2\cos\phi)^2)] \end{aligned} \tag{2.47}$$

2.3.3 Inductive interactions

The final, and weakest, polar interactions are inductive interactions. These occur when a non-polar molecule interacts with an ionic or polar species, resulting in an induced dipole on the former. The size of the induced dipole depends on both the magnitude of the inducing charge and the polarisability α_0 of the non-polar molecule.

Polarisation of the non-polar molecule occurs when the approaching charge draws electrons towards it (if positive) or repels them (if negative). This forces electron density to certain regions of the molecule and away from others, inducing a dipole. The induced dipole then interacts with the ionic/polar molecule. The energy of a charge-induced dipole interaction is

$$E_{\mathrm{CI}} = -\frac{Q^2\alpha_0}{(4\pi\epsilon_0)^2 r^4}, \tag{2.48}$$

while for a dipole-induced dipole interaction it is

$$E_{\mathrm{DI}} = -\frac{\mu^2\alpha_0(1 + 3\cos^2\theta)}{2(4\pi\epsilon_0)^2 r^6}. \tag{2.49}$$

These energies are lower than those of permanent charge-dipole and dipole-dipole interactions. The reasons for this are twofold: the further away the inducing molecule, the smaller the induced dipole and the weaker the interaction;

and some energy is lost to the rearrangement of electron density on the molecule being polarised. The former point is illustrated by the respective r^{-4} and r^{-6} dependencies of E_{CI} and E_{DI}.

2.3.4 Dispersion interactions

Aside from subsection 2.3.1, all intermolecular interactions considered thus far are contingent on the presence of polar molecules. As such, they are present only in certain interactions. Dispersion interactions, however, are present in all intermolecular interactions.

Dispersion interactions arise between molecules due to the development of instantaneous dipoles. Though a molecule may have no net dipole, the constant flux in electron density leads to the development of instantaneous dipoles. These then induce a dipole on any proximal molecule. Dispersion interactions are, in effect, temporary dipole-induced dipole interactions.

For two identical molecules, the dispersion interaction energy is approximately

$$E_{\mathrm{Disp}} \approx -\frac{3\alpha_0^2 I}{4(4\pi\epsilon_0)^2 r^6} = -\frac{\mathrm{C}_6}{r^6}. \tag{2.50}$$

Here I is the ionisation energy of the molecule and the C_6 term is the dispersion coefficient for the molecule, which is always positive. Equation (2.50), therefore, shows that dispersion interactions are always stabilising. It also illustrates that $E_{\mathrm{Disp}} \propto r^{-6}$, just as for E_{DI} in the previous section.

Approximately, the energy of the dispersion interactions between non-identical molecules is

$$E_{\mathrm{Disp}} \approx -\frac{3\alpha_0^{(1)}\alpha_0^{(2)}}{2(4\pi\epsilon_0)^2 r^6}\frac{I_1 I_2}{I_1 + I_2}, \tag{2.51}$$

where $\alpha_0^{(i)}$ and I_i are the polarisability and ionisation energy of molecule i respectively. Once again, equation (2.51) illustrates the energy of the stabilising dispersion interactions possesses an r^{-6} dependence.

2.3.5 The supermolecule approach

The total potential energy U of a system of N_{m} molecules is [118]

$$U = \sum_{i}^{N_{\mathrm{m}}} E_{1\mathrm{B}}(\mathbf{p}_i) + \sum_{i<j}^{N_{\mathrm{m}}} E_{2\mathrm{B}}(\mathbf{p}_i, \mathbf{p}_j) + \sum_{i<j<k}^{N_{\mathrm{m}}} E_{3\mathrm{B}}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k) + \ldots + E_{N_{\mathrm{m}}B}, \tag{2.52}$$

where E_{ZB} is the Z-body energy of interaction between Z molecules and $\mathbf{p}_i$ is the position of molecule i. The one-body energies are the energies of the individual molecules, the two-body energies those of the pairwise intermolecular interactions and so on. As Z increases, the contribution of the Z-body energy to U typically decreases.

The one-body terms account for the individual energies of all the molecules, while the higher order terms cover the intermolecular interaction energy U_{int}. The value of U_{int} depends on the type and number of the previously-discussed interactions present. U_{int} can be calculated via the supermolecule approach, which for a system of two atoms gives [119]

$$U_{\text{int}} = U - E_{1\text{B}}(\mathbf{p}_1) - E_{1\text{B}}(\mathbf{p}_2). \tag{2.53}$$

This approach requires accurate *ab initio* calculations to be used because U_{int} is far smaller than the total one-body energy. These calculations make use of the Born-Oppenheimer approximation, meaning only the electronic wavefunctions Ψ_{el} must be approximated. Furthermore, the supermolecule approach introduces a basis set superposition error. This error derives from the finite number of basis functions used: as one molecule approaches another, the basis functions from one stabilise the other and *vice versa*. This leads to an overestimate of U_{int}, which is more pronounced with smaller basis sets.

The basis set superposition error can be rectified by the counterpoise correction [120]. For $N_{\text{m}} = 2$, the non-corrected supermolecular interaction energy is

$$U_{\text{int}} = U^{(\text{AB})}(\mathbf{p}_1, \mathbf{p}_2) - E_{1\text{B}}^{(\text{A})}(\mathbf{p}_1) - E_{1\text{B}}^{(\text{B})}(\mathbf{p}_2), \tag{2.54}$$

where the superscripts denote the basis sets. Under the counterpoise correction, the basis sets of molecule 2 are included when the energy of molecule 1 is determined. These are centred on the nuclear coordinates of molecule 2, but do not include its electrons or nuclear charge. That is, the calculation of E_1 is done in the presence of a 'dummy' molecule 2. The same procedure is followed for evaluating the energy of molecule 2, giving the counterpoise-corrected energy as

$$U_{\text{int}} = U^{(\text{AB})}(\mathbf{p}_1, \mathbf{p}_2) - E_{1\text{B}}^{(\text{AB})}(\mathbf{p}_1) - E_{1\text{B}}^{(\text{AB})}(\mathbf{p}_2). \tag{2.55}$$

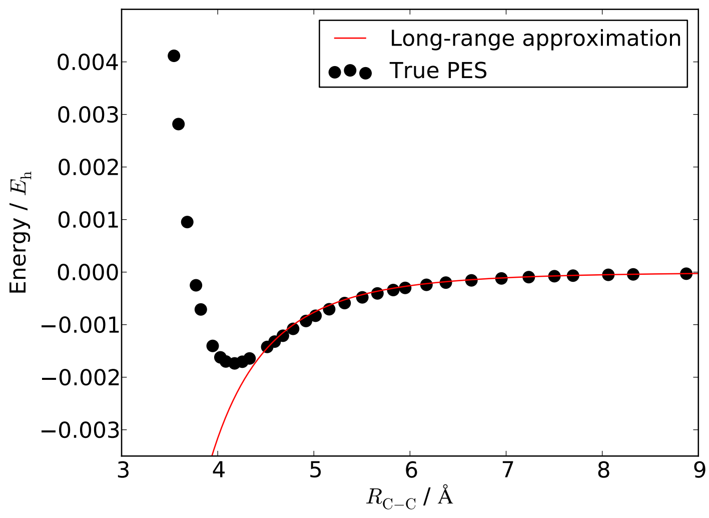


Figure 2.4: Plot showing the divergence of the multipole expansion (red line) from calculated data on the $(CO_2)_2$ potential energy surface. The data are from MP2 calculations with the molecules in a T-shaped configuration.

2.3.6 Additive and non-additive interactions

The two-body contributions to U_int are described as the additive contributions. This is because their total amounts to adding all of the pairwise interactions between the molecules. All higher order terms are the non-additive terms. It has been shown that the two- and three-body terms account for over 95 % of the the total intermolecular interaction energy of water [121–124], so the highest order non-additive contribution explored here is the three-body energy.

The three-body energy is explained by considering three interacting molecules, labelled 1, 2 and 3. The intermolecular interaction energy is given by

$$U_\mathrm{int} = E_\mathrm{2B}(\mathbf{p}_1, \mathbf{p}_2) + E_\mathrm{2B}(\mathbf{p}_1, \mathbf{p}_3) + E_\mathrm{2B}(\mathbf{p}_2, \mathbf{p}_3) + E_\mathrm{3B}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3), \tag{2.56}$$

where the first three terms are the two-body energies and the final term the three-body energy. The latter is referred to as non-additive as it accounts for the difference between U_int and the sum of the two-body, additive energies. It arises because the 1-2 interaction will be affected by polarisation of 1 and 2 by 3, with the same true of 1 and 3 by 2, and 2 and 3 by 1.

2.3.7 The multipole expansion

For a system of two small molecules at centre-centre separation r, the interaction energy can be approximated using the multipole expansion. This expansion estimates the interaction energy as a Taylor series in $1/r$ about the point $1/r = 0$. Equations (2.43)-(2.51) derive from this expansion and all multipolar long-range functions in Chapter 3 are obtained by summing the relevant contributions from these equations. Though the full expansion comprises theoretically an infinite number of terms, all of these long-range functions include terms up to r^{-6} only.

There are two sources of error in a multipole expansion: penetration and truncation. The latter is a consequence of approximating the expansion to a finite number of terms, which can cause convergence to an erroneous energy at long range. However, it is often possible to determine the appropriate power of r at which to truncate the series for a given interaction. The penetration error, meanwhile, is a consequence of short-range repulsion between the electron clouds of the interacting molecules. Below a certain separation this effect dominates the long-range interactions discussed previously, leading to divergence of the multipole expansion from the true energy. The short-range divergence of the multipole expansion is illustrated in figure 2.4 and is the reason that the functions that result from the multipole expansion are referred to as long-range functions here.

Below the separation at which divergence begins, it is paramount to cross-over to another method of prediction such as a Gaussian process. Hence this distance is analogous to the cross-over distance R_{cross} mentioned in section 2.1.4. At $r > R_{\text{cross}}$, a long-range function that converges appropriately is an excellent method of prediction for three reasons: it is cheap to evaluate; it will make accurate, physically motivated predictions of the long-range interaction energy; and its form as a power law makes it easy to integrate when evaluating long-range corrections. Thus, choosing R_{cross} appropriately not only reduces training set size, it permits predictions to be made by long-range functions. This reduces overall computational expense without affecting accuracy. In Chapter 3, long-range functions are assumed to converge if they exhibit good agreement with the long-range reference data, where "good agreement" is defined as sufficiently accurate that the GP is not preferred for prediction easily.

2.4 Statistical mechanics and molecular simulations

Knowledge of the potential energy of a group of molecules as a function of their positions imbues understanding of the microscopic behaviour of those molecules. However, translating this to bulk properties is not necessarily straightforward. Molecular simulations are powerful tools for estimating the thermophysical properties of bulk materials from the interactions between the constituent molecules. These estimates are made using statistical mechanics, which links microscopic interactions to macroscopic observable properties. For any property, the simulation method in effect generates different samples for which the property is evaluated using statistical mechanics. Two popular molecular simulation methods are Monte Carlo and molecular dynamics simulations. The former proceeds by moving particles at random while accepting or rejecting moves based on the change in energy. In molecular dynamics, meanwhile, the movement of any particle is informed by the forces acting upon it.

As alluded to in Chapter 1, a prospective implementation of the methods outlined in Chapters 3-5 is the determination of phase transitions in carbon capture and storage pipelines. This can be achieved by a Monte Carlo simulation that uses an appropriate ensemble, so the focus of this discussion is Monte Carlo simulation. This method has seen recent applications ranging from evaluating the thermodynamic properties of ferroelectric materials [125] to assessing the use of metallic nanoparticles in reducing side-effects during radiotherapy [126].

The section begins with an overview of statistical mechanics and sampling with ensembles. This precedes discussions of Monte Carlo simulations and determination of phase transitions within these simulations. The section concludes by outlining some common practices in molecular simulations. This discussion follows 'Computer Simulations of Liquids' by Allen and Tildesley (Chapters 1, 2 and 4) [127] and 'Molecular Simulations of Fluids' by Sadus (Chapters 2, 5, 8 and 9) [128] closely. Its goal is to introduce the concepts in molecular simulation that motivate the work in Chapter 5.

2.4.1 Statistical mechanics and ensembles

In statistical mechanics, the observed value, A_{obs}, of some macroscopic property A of a system is determined by its average value over a sufficient period of time t. That is, for a system at equilibrium,

$$A_{\mathrm{obs}} = \langle A(\Gamma) \rangle_{\mathrm{t}}. \tag{2.57}$$

At any time, the value of A is a function of the positions and momenta of the particles of the system. These can be envisioned as coordinates in a multidimensional space called phase space, which has $6N$ dimensions for a system of N particles, with three dimensions each for the position and momentum coordinates of each particle.

However, evaluation of the time average is complex for large numbers of molecules, even when discretised into time-steps. Instead, time averages are replaced by ensemble averages, where an ensemble is a collection of microstates from phase space. Each microstate shares a set of macroscopic properties that are fixed during the simulation. Examples of properties that are fixed in an ensemble include the energy E, pressure P, volume V, number of particles N and temperature T.

The use of ensemble averages is predicated on the assumption that the time average and ensemble average are equivalent, which is known as the ergodic hypothesis. This states that

$$\langle A \rangle_{\mathrm{t}=\infty} = \langle A \rangle_{\mathrm{ens}}, \tag{2.58}$$

which is to say that the ensemble average equates to the observed value of A over infinite time. This holds for ergodic systems, in which all configurations with a non-zero probability function can be explored during the simulation.

The ensemble average of A is taken by averaging its value over microstates that are samples from phase space,

$$\langle A \rangle_{\mathrm{ens}} = \sum_i A_i p_i. \tag{2.59}$$

Here A_i is the value of A in the ith microstate and p_i is the probability distribution associated with the ensemble. For an ensemble in which the number of particles, volume and temperature are constant (*i.e.* a constant-NVT ensemble) p_i is based on the Boltzmann equation,

$$p_i = \frac{\exp(-\frac{E_i}{k_\mathrm{B}T})}{Z_\mathrm{NVT}}, \tag{2.60}$$

where k_B is the Boltzmann factor and Z_NVT is the partition function for the ensemble,

$$Z_\mathrm{NVT} = \sum_i \exp\Big(-\frac{E_i}{k_\mathrm{B}T}\Big). \tag{2.61}$$

When employing ensembles, a microstate evolves over simulation steps. Exactly how the system evolves at each step is dependent on the simulation technique employed. For molecular dynamics, all particles move under the forces exerted on them over a certain duration, rendering each step a time-step. In Monte Carlo simulations, which have no time-dependence, each step comprises one of many possible moves, with the simplest being the displacement of a single particle over a random distance and direction.

2.4.2 Monte Carlo simulations

Monte Carlo (MC) simulations are probabilistic strategies that sample equilibrium distributions of systems to obtain thermophysical properties as equilibrium averages. MC simulations do not depend on the momenta of the particles, meaning their microstates are taken from a $3N$-dimensional phase space. This discrepancy with molecular dynamics is not an issue, however, as thermodynamic properties of mixtures do not depend on the momenta of the particles [128].

Finding the average of a property over the microstates is equivalent to evaluating

$$\langle A \rangle = \int A(\overline{\mathbf{p}})p(\overline{\mathbf{p}})d\overline{\mathbf{p}}. \tag{2.62}$$

Here $\overline{\mathbf{p}}$ is a N x 3 matrix of the particle coordinates and $p(\overline{\mathbf{p}})$ is a continuous equivalent to the ensemble probability distributions discussed in the last section,

$$p(\overline{\mathbf{p}}) = \frac{\exp(-\frac{E(\overline{\mathbf{p}})}{k_\mathrm{B}T})}{\int \exp(-\frac{E(\overline{\mathbf{p}})}{k_\mathrm{B}T})d\overline{\mathbf{p}}}. \tag{2.63}$$

However, solving these integrals either analytically or numerically is not feasible. The Monte Carlo solution is to generate N_trial sample microstates by moving molecules randomly, replacing the integral in equation (2.62) with the sum

$$\langle A\rangle = \frac{\sum_i^{N_{\rm trial}} A_i \exp(-\frac{E_i}{k_{\rm B}T})}{\sum_i^{N_{\rm trial}} \exp(-\frac{E_i}{k_{\rm B}T})}. \tag{2.64}$$

This is consistent with applying the constant-NVT probability function to equation (2.59), though could be altered to include the constant-NPT probability easily.

Though more tractable than the integral, for random sampling the requisite value of $N_{\rm trial}$ is prohibitively large regardless of ensemble. This is because truly random sampling will include many more low-probability microstates than those that make a significant contribution to $\langle A\rangle$. This issue is exacerbated as the number of particles increases. Consequently, importance sampling is used to ensure microstates from high-probability regions of phase space are favoured.

Importance sampling proceeds by envisioning the evolution of a microstate from step τ to ω as a Markov chain. That is, the microstate at step ω defined by the positions $\overline{\mathbf{p}}(\omega)$ depends only on the previous microstate $\overline{\mathbf{p}}(\tau)$. For every τ there are K possible $\tau \rightarrow \omega$ transitions, each with an associated transition probability $\pi_{\tau\rightarrow\omega}$. It is specified that

$$\sum^{K} \pi_{\tau\rightarrow\omega} = 1, \tag{2.65}$$

where K is the number of microstates to which transitions are possible from $\overline{\mathbf{p}}(\tau)$. This ensures that every adjacent microstate can be reached. The value of $\pi_{\tau\rightarrow\omega}$ for any transition is

$$\pi_{\tau\rightarrow\omega} = \begin{cases} C_{\tau\rightarrow\omega}, & \frac{p_\omega}{p_\tau} \geq 1 \\ C_{\tau\rightarrow\omega}\frac{p_\omega}{p_\tau}, & \frac{p_\omega}{p_\tau} < 1, \end{cases} \tag{2.66}$$

where $C_{\tau\rightarrow\omega}$ is the probability of picking a given $\overline{\mathbf{p}}(\omega)$ from all possible microstates. As it is possible to visit each of the K adjacent microstates $C_{\tau\rightarrow\omega}K = 1$, meaning $C_{\tau\rightarrow\omega} = 1/K$.

Equation (2.66) shows that whether to transition from $\overline{\mathbf{p}}(\tau)$ to $\overline{\mathbf{p}}(\omega)$ is determined by evaluating p_ω/p_τ, where both probabilities come from equation (2.63). As the ratio of the two probabilities is being determined, the integral from equation (2.63) need not be solved and for a constant-NVT ensemble

$$\frac{p_\omega}{p_\tau} = \frac{\exp(-\frac{E_\omega}{k_{\rm B}T})}{\exp(-\frac{E_\tau}{k_{\rm B}T})} = \exp\Big(-\frac{\Delta E}{k_{\rm B}T}\Big). \tag{2.67}$$

In equation (2.67), $\exp\Big(-\frac{\Delta E}{k_\mathrm{B}T}\Big) > 1$ if state $\overline{\mathbf{p}}(\omega)$ has a lower energy. In such a case, the $\overline{\mathbf{p}}(\tau) \rightarrow \overline{\mathbf{p}}(\omega)$ transition is accepted immediately as it leads to a more probable region of phase space. If the energy at $\overline{\mathbf{p}}(\omega)$ exceeds that of the previous state however, the move can be accepted if $\exp\Big(-\frac{\Delta E}{k_\mathrm{B}T}\Big) \geq S$ where S is uniformly distributed on [0,1]. This is known as the Metropolis method and is characterised by defining the probability of acceptance as

$$p = \min\Big[1, \exp\Big(-\frac{\Delta E}{k_\mathrm{B}T}\Big)\Big] \tag{2.68}$$

for a constant-NVT ensemble. Though this may lead to moves away from regions of high probability, a sequence of poor moves is unlikely and the acceptance of moves that increase energy reduces the chance of being trapped in local minima.

However, by influencing the regions of phase space explored during the simulation, importance sampling introduces bias. This bias is accounted for by imparting a weight, w_i, on the ith microstate. Equation (2.64) then becomes

$$\langle A \rangle = \frac{\sum_i^{N_\mathrm{trial}} A_i \exp(-\frac{E_i}{k_\mathrm{B}T})/w_i}{\sum_i^{N_\mathrm{trial}} \exp(-\frac{E_i}{k_\mathrm{B}T})/w_i}. \tag{2.69}$$

Choosing $w_i = \exp(-\frac{E_i}{k_\mathrm{B}T})$ this simplifies to

$$\langle A \rangle = \frac{1}{N_\mathrm{trial}} \sum_{i=1}^{N_\mathrm{trial}} A_i \tag{2.70}$$

for a constant-NVT ensemble. Thus by choosing the Boltzmann factor as the weight for a constant-NVT sample, $\langle A \rangle$ is simply the average of A from all microstates. Equivalent results are achieved for other ensembles by modifying the numerator in the exponent of the Boltzmann factor (*i.e.* the E_i term).

As the energy must be re-evaluated after any move, MC simulations are predicated on multiple energy evaluations. Consequently, the evaluation of the energy determines the computational cost of any MC simulation. Thus the use of a potential that is more intensive computationally than another for a single energy evaluation will increase appreciably the duration of the simulation, which will necessitate thousands of these calculations at every step. Put simply, the total CPU time of any simulation will be proportional to the CPU time of evaluating the potential. As a result, minimising the cost of using GP potentials in such simulations is crucial.

2.4.3 Phase coexistence in Monte Carlo simulations

Phase coexistence in molecular simulation requires equivalence of temperature, pressure and chemical potential η between the two phases. Respectively, equivalence of these quantities corresponds to thermal equilibrium, mechanical equilibrium and chemical equilibrium. This leads intuitively to the use of a constant-ηPT ensemble. However, it transpires that such an ensemble is a rather inconvenient way to simulate phase coexistence practically. Instead, the Gibbs ensemble [129, 130] is employed commonly for modelling such behaviour. In the context of the work in later chapters, the Gibbs ensemble would be used to simulate the vapour-liquid coexistence point as a function of temperature and pressure.

The Gibbs ensemble separates the system being modelled into two distinct regions. In this discussion, one region is a liquid and the other gaseous. The ensemble attains thermal equilibrium through random movement of particles (displacement) and mechanical equilibrium via fluctuations in the volume of each region (volume changes). Chemical equilibrium is achieved by allowing particles to exchange between both regions (particle exchange). Displacement, volume change and particle exchanges are accepted with a probability of

$$p = \min\Big[1, \exp\Big(-\frac{\Delta Y}{k_{\mathrm{B}}T}\Big)\Big]. \tag{2.71}$$

This is a generalisation of equation (2.68) where ΔY differs depending on which of the three moves is made (see Appendix C).

Displacements, which occur in almost all Monte Carlo simulations, are the most common of the three moves and entail movement and rotation of a random particle. For displacements the change in energy arises from the interactions of the moved particle, which means only these must be re-calculated. Typically, an acceptance rate of 50 % achieves thermal equilibrium efficiently. This rate is altered by adjusting the maximum possible displacement in the simulation.

Exchange moves are similar, with the change in energy of the region into which the particle was transferred altered only by the interactions of that particle. However, exchanges have the additional requirement that the energy in the region from which the particle was taken be re-calculated to no longer include it. Exchange moves also require a far lower acceptance rate of only a few percent to achieve chemical equilibrium. Attainment of chemical equilibrium is verifiable through calculation of the chemical potential in both regions.

Volume changes, which are also employed in constant-NPT simulations, require complete re-calculation of the energy as the intermolecular separations are scaled by the same factor as the side length. As such, they are the most computationally intensive of the moves required to simulate phase co-existence and must be handled efficiently. Like displacements, volume changes require an acceptance rate of ~ 50 % to achieve material equilibrium. The rate is adjusted by altering the maximum allowed fluctuation in side length. Both volume changes and exchanges are attempted with less regularity than displacements.

A Gibbs ensemble can run as either constant-NVT or constant-NPT, with the difference in the treatment of the volume changes. Under the former it is specified that the total change in volume over the two regions is zero to ensure that there is no net change in volume. Meanwhile, a constant-NPT Gibbs ensemble treats the volume fluctuations as completely independent.

2.4.4 Common practices in molecular simulations

Periodic boundary conditions

Consider a simulation of liquid argon for which the number of atoms, N_a, is 1000 in a cubic simulation box with side length $L = 10$. Each atom could be placed in either the bulk of the liquid or on its surface. Whereas in this simulation box the proportion of atoms at the surface will be quite large, the same is not true of a real liquid for which surface effects are not so prominent. Because the interactions undergone by surface atoms will differ greatly from those in the bulk, any physically reasonable simulation has to address surface effects.

The issue of surface effects is circumvented by the introduction of periodic boundary conditions (PBCs). Under PBCs, the aforementioned simulation box is replicated throughout space to form an infinite lattice. Thus one can consider a simulation that employs PBCs to comprise an infinite number of simulation boxes. Each simulation box contains N_a atoms indexed as a_b, where a is the atom index and b denotes its simulation box.

When atom a_1 moves, all $a_{b\neq 1}$ move identically within their own simulation boxes. The borders of the simulation boxes do not represent physical barriers because PBCs are implemented to mitigate surface effects, meaning it is feasible that any atom could leave its original simulation box through one of its faces. In such a case, the equivalent atom from an adjacent simulation box will replace it by entering through the opposite face. The consequences of this are twofold:

- N_a remains constant for each simulation box throughout the simulation, excluding particle exchange moves.
- Only the positions of the atoms in the original simulation box need to be stored.

This leaves the question of whether the properties of a relatively small, infinitely repeating simulation box can reflect accurately those of a true bulk liquid. The answer is dependent on two things: the range of the intermolecular potential employed in the simulation and the property being investigated.

For most properties, PBCs do not present a problem [127], hence their widespread use in simulations. However, capturing properties close to the critical point is problematic. This is because the correlation length of any change to the bulk near to the critical point will exceed L. That is, in a real liquid at the critical point, two atoms that are more than a side length apart will be correlated. This is potentially problematic as in a simulation atoms at separations beyond L are correlated incorrectly because they are images of each other. Though one could increase L massively to account for this, such a change could render the simulation prohibitively expensive.

The range of the potential, meanwhile, plays a key role because a true liquid is not a series of interconnected, repeating boxes. Thus a potential that allows a particle to interact explicitly with its image in an adjacent box will reduce simulation accuracy by capturing the unphysical nature of the PBCs. Therefore, for any simulation that employs PBCs to capture the properties of a bulk liquid it must be ensured that atom a_1 does not interact explicitly with any equivalent atom $a_{b\neq 1}$.

Minimum image convention and cut-off distances

The issue of preventing a given atom from interacting with images of itself is circumvented by application of a minimum image convention (MIC). An MIC entails placing the atom for which the potential is being calculated at the centre of a box with the same dimensions as the original simulation box. Only the interactions of this atom with all others for which the centres lie within the minimum image box are considered. As any images of the central atom will lie outside of this box, its interaction with them is not considered explicitly.

An MIC also accounts for another issue introduced by the PBCs: that there are an infinite number of simulation boxes repeated throughout space. Both Monte Carlo and molecular dynamics simulations are reliant upon calculations of the total potential energy of the system being studied and, in the latter case, the forces acting on all atoms. Consider the additive contribution to the potential energy of the interactions involving atom a_1, $U_{\text{Add}}^{(a_b)}$. For a single simulation box this would be

$$U_{\text{Add}}^{(a_1)} = \sum_{i<a} U_{\text{Add}}^{(a_1 i_1)} + \sum_{j>a} U_{\text{Add}}^{(a_1 j_1)}, \tag{2.72}$$

a tractable summation containing $N_{\text{a}} - 1$ terms.

However, under PBCs the interactions of a_1 with all atoms in all other simulation boxes must also be evaluated. Such a summation is impossible as it contains an infinite number of terms. Under an MIC, the potential of each atom is given by a sum over the $N_{\text{a}} - 1$ closest periodic images of the other atoms in the simulation. Thus, calculation of the total additive energy involves $(N_{\text{a}}^2 - N_{\text{a}})/2$ terms under a MIC.

For systems where N_{a} is big, $(N_{\text{a}}^2 - N_{\text{a}})/2$ could be prohibitively large. Thus, another simplification is made whereby a sphere with a cut-off radius of r_{c} is constructed around the atom being considered. Only the interactions of this atom and all others within this sphere are considered explicitly. For consistency with the MIC, r_{c} should not exceed $\frac{L}{2}$. When using machine-learned potentials (MLPs) for prediction a sensible choice is $r_{\text{c}} = R_{\text{cross}}$, the crossing point between the MLP and the long-range function. This holds only if $R_{\text{cross}} \leq \frac{L}{2}$, though selecting a smaller r_{c} in this case would still ensure the MLP is used in a region where its predictions are accurate.

Long-range corrections

For a potential comprising short-range interactions, the explicit additive energy of any interaction between atoms at separation $r > r_{\mathrm{c}}$ is considered to be zero. This is because the total additive potential is dominated by the contributions between atoms at low separations. For long-range interactions, a long-range correction is applied. Examining the additive potential, such a correction views the total additive energy $U^{\mathrm{tot}}_{\mathrm{Add}}$ as

$$U^{\mathrm{tot}}_{\mathrm{Add}} = U^{\mathrm{c}}_{\mathrm{Add}} + U^{\mathrm{LRC}}_{\mathrm{Add}}, \tag{2.73}$$

where $U^{\mathrm{c}}_{\mathrm{Add}}$ is the explicitly-calculated additive energy within the cut-off radius and $U^{\mathrm{LRC}}_{\mathrm{Add}}$ is the long-range correction. This correction assumes that particles separated by more than r_{c} from an atom a are uniformly distributed around this atom. Thus a interacts with a mean field of these atoms. Under this mean field assumption, the atomic $U^{\mathrm{LRC}}_{\mathrm{Add}}$ is given by [127, 128]

$$U^{\mathrm{LRC}}_{\mathrm{Add}} = 2\pi N_a \rho \int_{r_{\mathrm{c}}}^{\infty} r^2 E^{(2)} \mathrm{d}r, \tag{2.74}$$

where ρ is the density of the liquid and $E^{(2)}$ is a function that approximates the long-range additive interaction energies.

When considering a simulation of argon atoms, a sensible choice of $E^{(2)}$ is energy of the dispersion interactions,

$$E^{(2)} = -C_6 r^{-6}, \tag{2.75}$$

where C_6 is the dispersion coefficient for argon. Thus the additive long-range correction is

$$U^{\mathrm{LRC}}_{\mathrm{Add}} = -\frac{2}{3}\pi N_a \rho C_6 r_c^{-3}. \tag{2.76}$$

For constant N_a, ρ and r_{c} this quantity needs to be calculated once only. The application of the additive long-range correction to a simulation is therefore straightforward, although it must be re-calculated if N_{a} or ρ vary. These correspond respectively to exchange and volume change moves but involve altering these trivial terms only.

Meanwhile, though more complicated, three-body long-range corrections for atomic systems already exist. One example is that proposed by Leonhard and

Deiters [131], which gives the long-range correction as

$$U_{\mathrm{LD}}^{\mathrm{LRC}} = 8\pi^2 \frac{N_{\mathrm{a}}\rho^2}{3!} \int_{r_c}^{\infty} \int_{r_c}^{\infty} \int_{\max(r_c, |r_{12}-r_{23}|)}^{r_{12}+r_{13}} E^{(3)} dr_{12} dr_{13} dr_{23}, \tag{2.77}$$

where r_{12}, r_{13} and r_{23} are the three interatomic distances that comprise the triplet. Leonhard and Deiters used the Axilrod-Teller potential [132],

$$E^{(3)} = C_{\mathrm{AT}} \left(1 + \frac{3}{8} \frac{(r_{12}^2 + r_{13}^2 - r_{23}^2)(r_{13}^2 + r_{23}^2 - r_{12}^2)(r_{12}^2 + r_{23}^2 - r_{13}^2)}{r_{12}^2 r_{13}^2 r_{23}^2} \right) \frac{1}{r_{12}^2 r_{13}^2 r_{23}^2}. \tag{2.78}$$

Here $C_{\mathrm{AT}} \approx I\alpha^3$, where I is the ionisation constant of the atom and α its polarisability.

The drawbacks of this approach include that it does not account for the configurations in which one or two distances exceed r_{c}. Moreover, it is not applicable to systems of non-noble gases, as the Axilrod-Teller potential applies only to these species. However, the integral could be generalised by using alternative potentials in equation (2.77). Regardless of the form used, both the three- and two-body long-range corrections must be re-evaluated after a particle exchange or volume change move. These re-evaluations pertain respectively to the trivial N_{a} and ρ terms only, making this a simple procedure.

2.5 Summary of background

In summary, Gaussian process (GP) potentials are a promising choice for use in molecular simulations not only because they can deliver first principles predictions but because they employ a physically motivated function that can be integrated straightforwardly for the additive long-range correction. This is in addition to GP potentials outperforming other methods of prediction for a fixed training set size and evaluating energies faster than even the most efficient *ab initio* methods, at little cost to accuracy. Ideally, the region predicted by the GP would be as small as possible at fixed accuracy to minimise GP evaluations and increase speed. Simultaneously, a minimal training set is important to further increase speed and reduce the computational cost of upgrading to a higher level of theory.

The work in the following chapters outlines a method to reduce the training set size and, in many cases, cross-over distance for GP potentials. It also presents a method for exploiting the GP framework to implement GP potentials as efficiently as possible in simulations. Both are important considerations because, as stated, GP potentials are more computationally intensive than other potentials for fixed training set size, as well as being more intensive to evaluate than a long-range function. However, effective strategies to circumvent these issues permit the use of GP potentials in simulations, offering an excellent route to applying high-level *ab initio* information to the estimation of macroscopic properties of molecular mixtures.

Chapter 3

Gaussian process potentials with boundary optimisation

As discussed, it is paramount to reduce training set size for Gaussian process (GP) potentials to both increase speed of prediction and facilitate transfer learning. The methods discussed in Chapter 2.1 showed how the training sets for GP potentials were contracted by moving from Latin hypercube training to sequential design. However, in both cases the cross-over distance R_{cross} between the GP and long-range function was chosen *a priori*.

In this chapter, the highest error search method of Uteva *et al*. [29] is extended so that R_{cross} is no longer fixed in advance but learned from the reference data. This facilitates further reductions in training set size of up to 40 % at no cost to accuracy. In addition, the resultant R_{cross} is often smaller than the 8.5 Å chosen in the past. This means that a larger proportion of energy predictions in any simulation can be left to the long-range function, which is cheaper to evaluate than the GP, without reducing accuracy. This is achieved by boundary optimisation, which optimises R_{cross} each time the GP is updated using a direct search. This direct search is fast with respect to hyperparameter re-optimisation, meaning the time taken to train the GP is barely affected.

3.1 Motivation for boundary optimisation

As stated, the goal of boundary optimisation is to further decrease the number of training points N_t required to develop a GP potential of a given accuracy relative to the highest error search method of Uteva *et al.* [29], which was discussed in section 2.1.4. Under that method, R_{cross} was fixed form the outset at 8.5 Å. Recall also that the highest error search proceeded by adding the configuration in the reference set $\mathcal{R}$ for which the squared error in the GP prediction was highest to the training set $\mathcal{T}$. Meanwhile, $\mathcal{T}$ was instantiated with the highest energy point in $\mathcal{R}$. All configurations added to $\mathcal{T}$ were removed from $\mathcal{R}$.

It was discussed in subsection 2.3.7 that long-range functions from the multipole expansion are capable of accurate prediction even at separations just above the equilibrium bond length, as illustrated in figure 2.4. Such functions are used at long-range in the GP potentials of Uteva *et al.* [27]. Thus, at low N_t the predictions of the long-range function at the outer edge of the potential well are likely to surpass those of the GP in terms of accuracy. Consequently, the accuracy of the potential will be increased relative to a highest error search with fixed R_{cross}. This is because, by permitting optimisation of this quantity, the region predicted by the long-range function is maximised. Intuitively, it is anticipated that the associated increase in model accuracy will be greater at lower N_t.

Boundary optimisation may also enhance training efficiency by allowing the GP to gradually expand the region in which it makes predictions. Under a fixed R_{cross}, the GP is used for prediction at low separations in the repulsive wall and larger separations at the outer edge of the potential well simultaneously. As the rate of change in the energy with separation is different in both regions, so is the ideal lengthscale. This is circumvented under boundary optimisation, which allows the GP to learn the short-range region where the rate of change in the energy is highest first before expanding to larger separations. As no attempt has been made previously to learn R_{cross} from the reference data, any improvement facilitated by boundary optimisation is likely to transfer to previous methods of training machine-learned potentials, provided a suitable long-range function is available. Due to the speed of the direct search algorithm, this increase will accompany an at most negligible increase in training time.

3.2 Model specifications and data set design

All GPs discussed in this chapter make use of the symmetric squared exponential kernel [27]

$$
\begin{aligned}
k_{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j) &= \sigma_{\mathrm{f}}^2 \prod_{d=1}^{D} \exp\Big(- \frac{(x_i^{(d)} - x_j^{(d)})^2}{2 l_d^2} \Big) + \sigma_{\mathrm{n}}^2, \\
k_{\mathrm{Sym}}(\mathbf{x}_i, \mathbf{x}_j) &= \sum_{\mathrm{p} \in \mathrm{P}} k_{\mathrm{SE}}(\mathbf{x}_i, \mathrm{p}\mathbf{x}_j).
\end{aligned}
\tag{3.1}
$$

This was introduced in section 2.1.4, equation (2.22). The GPs were trained using the GPy package [133] in Python 2.7, with the hyperparameters re-optimised over 20 independent restarts whenever a configuration was added to $\mathcal{T}$. Moreover, a gamma distribution with an expectation of one and a variance of two was used as a prior on all hyperparameters for all systems. This was to weakly penalise large hyperparameter values, given that the expected values are typically of order 0.1 or below.

The systems for which GP potentials were developed with boundary optimisation are shown in table 3.1. Also shown in this table are the specifications of $\mathcal{R}$ and the test set $\mathcal{S}$ for each system. All data sets were produced under the LHC design discussed in section 2.1.4, only now $R_{\mathrm{cross}} \neq R_{\mathrm{max}} = 100$ Å. They were also subject to a high energy cut-off E_{cut}. This was 0.005 E_{h}[1] for all systems apart from HF-Na$^+$ ($E_{\mathrm{cut}} = 0.05 E_{\mathrm{h}}$) and $(\mathrm{HF})_2$ ($E_{\mathrm{cut}} = 0.002 E_{\mathrm{h}}$) as these systems have greater well-depths. All energies were calculated in Molpro [134] using MP2 with an aug-cc-pVTZ basis set and the counterpoise correction with the only exception being HF-Na$^+$, which used an aug-cc-pwCVTZ basis set instead (see Section 2.2.2).

In table 3.1 r is the distance between the bond centres (not centres of mass), θ is the angle between r and the bond axis of the molecule, θ_1 and θ_2 are the angles between r and the bond axis of the first and second $\mathrm{CO_2}$ molecules respectively, and ϕ is the torsional angle between the two $\mathrm{CO_2}$ molecules. The molecules were kept rigid, with $r_{\mathrm{CO}} = 1.1283$ Å for CO, $r_{\mathrm{CO}} = 1.1632$ Å for $\mathrm{CO_2}$ and $r_{\mathrm{HF}} = 0.9170$ Å. A larger geometric constraint of 100 Å (instead of 8.5 Å [27]) was employed to probe the long range behaviour of the system. The maximum value of N_{t} was 100 for all systems apart from $(\mathrm{CO_2})_2$, which used 300 training points

[1]1 $E_{\mathrm{h}} \approx 27.211$ eV ≈ 2625.5 kJ mol^{-1}.

Table 3.1: The co-ordinates for the reference and test LHCs for each system. N_{ref} and N_{test} are the number of points in the reference and test sets respectively after application of the high-energy cut-off, while the maximum number of training points for models of each potential are given in the text. Also shown is the minimum energy across the reference and test sets, E_{min}, in Hartrees (E_{h}), though no attempt was made to approximate the global minimum energy for any potential.

System	Coordinate	Range	N_{ref}	N_{test}	E_{min}
CO-Ne	r^{-1}	0.01 to 0.67 Å^{-1}	1914	5718	-1.502 x 10^{-4}
	$\cos(\theta)$	-1 to 1			
HF-Ne	r^{-1}	0.01 to 0.67 Å^{-1}	2148	6468	-2.633 x 10^{-4}
	$\cos(\theta)$	-1 to 1			
HF-Na$^+$	r^{-1}	0.01 to 0.67 Å^{-1}	2760	8416	-2.518 x 10^{-2}
	$\cos(\theta)$	-1 to 1			
CO_2-Ne	r^{-1}	0.01 to 0.67 Å^{-1}	5057	5072	-2.895 x 10^{-3}
	$\cos(\theta)$	0 to 1			
	r^{-1}	0.01 to 0.67 Å^{-1}	5810	5837	-1.975 x 10^{-3}
$(CO_2)_2$	$\cos(\theta_1)$	0 to 1			
	$\cos(\theta_2)$	0 to 1			
	ϕ	0 to 180$^{\text{o}}$			
	r^{-1}	0.01 to 0.67 Å^{-1}	7816	7839	-6.575 x 10^{-3}
$(HF)_2$	$\cos(\theta_1)$	-1 to 1			
	$\cos(\theta_2)$	-1 to 1			
	ϕ	0 to 180$^{\text{o}}$			
	r^{-1}	0.01 to 0.67 Å^{-1}	5759	1919	-3.125 x 10^{-3}
$(HCl)_2$	$\cos(\theta_1)$	-1 to 1			
	$\cos(\theta_2)$	-1 to 1			
	ϕ	0 to 180$^{\text{o}}$			

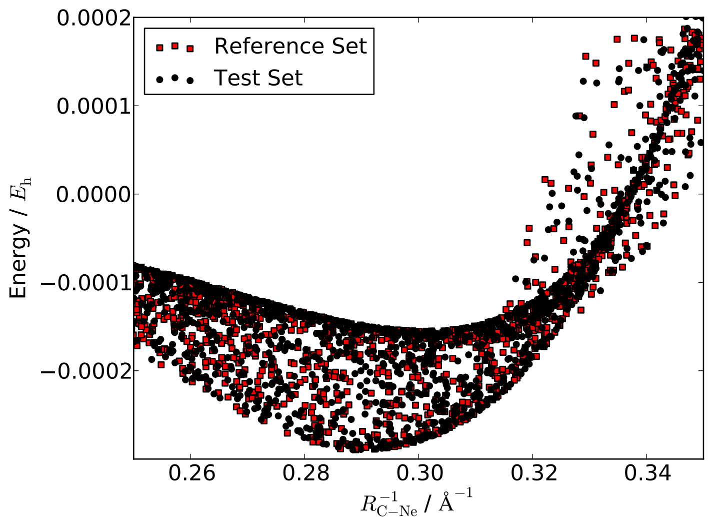


Figure 3.1: The calculated energies in the reference set (red) and the test set (black) of the CO_2-Ne potential for separations between 2.85 Å and 4 Å. Both sets contain $\sim$ 5000 configurations (see table 3.1).

at most, and the $(HX)_2$ dimers, which had $N_t = 350$.

For systems with $N_{\mathrm{test}} \gg N_{\mathrm{ref}}$ the independence of the test set is self-evident. However, for systems where $N_{\mathrm{test}} \approx N_{\mathrm{ref}}$ it is also true that the test set is independent. This follows because, although the reference and test sets for each system were designed using the same 'maximin' strategy, the stochastic nature of the LHC algorithm means that separate LHCs contain completely independent sets of configurations. Furthermore, the 'maximin' criterion is based on just one separation in the whole data set, meaning that two LHCs with similar maximin will still be dissimilar. This is demonstrated in figure 3.1, which shows the energies against the inverse C-Ne separation for the reference and test sets used in training models of the CO_2-Ne potential.

For the long-range energy model, multipole series were employed for all systems apart from HF-Na^+. The contributions included in the multipolar long-range functions are shown in table 3.2 for each system apart from CO_2-Ne, for which the function is already described in previous work [27], and $(CO_2)_2$. The latter was developed prior to the other multipolar long-range functions and uses atomistic charge, dipole, quadrupole and polarizability contributions from Hartree-Fock [109] theory, which were scaled to give the known total molecular properties. For HF-Na^+ a fitted long-range function was used, which was derived

Table 3.2: The properties included in the multipolar long-range functions of each system, as well as the *ab initio* methods used in their calculation. All calculations were carried out using an aug-cc-pVQZ basis set apart from those for the dispersion coefficients, which used an aug-cc-pVTZ basis set.

System	Dipole	Quadrupole	Polarizability	Dispersion
CO-Ne	✓	X	✓	✓
HF-Ne	✓	X	✓	✓
HF-Na^+	✓	✓	✓	✓
$(HX)_2$	✓	✓	✓	✓
Level of Theory	MRCI [135, 136]	MRCI	MP2	CCSD

by fitting an empirical sum of two power laws between two points where the energy was predicted by GP regression. More information on the motivation for and derivation of this function is found in appendix D.

3.3 Classification of input space with a boundary

When modelling potential energy surfaces (PESs) using GPs it is necessary to classify configurations as suitable for prediction via either the GP or a long-range function. In the work of Uteva *et al.* [27,29] the classifier formed a boundary from the superposition of atom-centred spheres defined by a single constant, R_{cross}. Specifically, if any interatomic distance was less than R_{cross} the GP was used. As R_{cross} was fixed, this classifier is referred to here as C_{fixed}. Denoting the region in which GP regression was used for prediction as $\mathcal{A}_{\text{GP}}$ and the region that employed the long-range function as $\mathcal{A}_{\text{LR}}$, under C_{fixed} these regions were

$$\mathcal{A}_{\text{GP}} = \{\mathbf{r} : \min(\mathbf{r}) \leq R_{\text{cross}}\} \tag{3.2}$$

and

$$\mathcal{A}_{\text{LR}} = \{\mathbf{r} : \min(\mathbf{r}) > R_{\text{cross}}\}, \tag{3.3}$$

where $\mathbf{r}$ is a set of intermolecular atom-atom distances, and $\min(\mathbf{r})$ is the smallest separation in $\mathbf{r}$.

In boundary optimisation, R_{cross} varies according to the GP accuracy. Thus R_{cross} is not constant but a model parameter, which is learned from the reference data. The classifier parameterised by a single, variable R_{cross} is referred to here as $\mathrm{C}_{\text{single}}$. Under $\mathrm{C}_{\text{single}}$, both regions are defined as above but are no longer fixed. That is, $\mathrm{C}_{\text{single}}$ is parameterised similarly to $\mathrm{C}_{\text{fixed}}$, only the variability of R_{cross} allows the sizes of $\mathcal{A}_{\text{GP}}$ and $\mathcal{A}_{\text{LR}}$ to vary under the former.

More elaborate classifiers are possible by using more detailed parametric forms to define the boundary region. A simple way of defining a more complex classifier is for the value of R_{cross} to depend on the atom types that comprise the interatomic distance. The resulting classifier is referred to here as $\mathrm{C}_{\text{multi}}$. For a system of molecules with D interatomic pairs of chemically different atoms, using $\mathrm{C}_{\text{multi}}$ requires the vector of cross-over distances $\mathbf{R}_{\text{cross}} = (R_1, ..., R_d, ..., R_D)$. This defines a multiple-parameter boundary region,

$$\mathrm{C}_{\text{multi}}(\mathbf{r}) = \begin{cases} \mathcal{A}_{\text{GP}}, & \text{if } \min_d(\mathbf{r}) \leq R_d \text{ for any } d \\ \mathcal{A}_{\text{LR}}, & \text{if } \min_d(\mathbf{r}) > R_d \text{ for all } d, \end{cases} \tag{3.4}$$

where $\min_d(\mathbf{r})$ is the minimum separation in $\mathbf{r}$ that involves an atomic pair of type d.

Optimal values of the classifier parameters are determined by minimising the error between the potential and the reference set (*i.e.* by minimising the training error), meaning the sizes of $\mathcal{A}_{\text{GP}}$ and $\mathcal{A}_{\text{LR}}$ vary with the GP. The sum of squared errors, $\mathrm{SSE}_{\text{tot}}$, over the two regions is

$$\mathrm{SSE}_{\text{tot}} = \mathrm{SSE}_{\text{GP}} + \mathrm{SSE}_{\text{LR}}, \tag{3.5}$$

where

$$\mathrm{SSE}_{\text{method}} = \sum_{i=1}^{N_{\text{method}}} (\hat{\mathrm{Y}}_i - \mathrm{Y}_i)^2. \tag{3.6}$$

Here,“method” denotes either GP or LR, N_{method} the number of points in $\mathcal{A}_{\text{method}}$, $\hat{\mathrm{Y}}_i$ the prediction of the energy for the ith configuration from the desired method and Y_i the calculated energy of the same configuration.

The RMSE against the test set, $\mathrm{RMSE}_{\mathrm{test}}$, is given by

$$\mathrm{RMSE}_{\mathrm{test}} = \left(\frac{\mathrm{SSE}_{\mathrm{test}}}{N_{\mathrm{test}}}\right)^{\frac{1}{2}}, \tag{3.7}$$

where $\mathrm{SSE}_{\mathrm{test}}$ is $\mathrm{SSE}_{\mathrm{tot}}$ over the test set and N_{test} is the number of configurations in the test set. $\mathrm{RMSE}_{\mathrm{test}}$ is therefore a function of R_{cross} with discrete steps, as the RMSE changes only when a variation in R_{cross} causes re-classification of a test configuration.

Both $\mathrm{C}_{\mathrm{single}}$ and $\mathrm{C}_{\mathrm{multi}}$ are simple, parametric classifiers that pre-impose a mathematical form on the classification. Hence, neither is expected to be optimal with respect to the RMSE against the reference or test sets. That is, a more complicated boundary than that described by these classifiers will likely produce a lower RMSE against a given data set. However, it is shown later that an artificial 'ideal' classifier produces only very marginal improvements over $\mathrm{C}_{\mathrm{multi}}$, suggesting this classifier balances simplicity and accuracy excellently.

A training strategy for a GP potential combines a classifier and a point placement strategy to build the potential. Using the data and classification methods above, a general training strategy to produce a GP potential sequentially proceeds as follows: train the GP to the current training set; select the classifier parameters by minimising the RMSE against the reference set; move a new point from the reference to the training set based on the largest error. Each step is elaborated upon below, along with how the choice of classifier and placement strategy affects the training strategy.

3.4 Direct search algorithm

When using $\mathrm{C}_{\mathrm{single}}$, R_{cross} is optimised via a direct search that exploits how the RMSE varies as a piecewise constant function of R_{cross}. Because of this feature, all possible values of the RMSE can readily be computed and a direct search of these values is guaranteed to find the global minimum. Full details of the direct search are given in algorithm 1.

Each instance of the direct search requires only a single set of predictions from each of the long range function and GP regression. As the long range function is fixed throughout the sequential design process, these predictions need only be

Algorithm 1 Direct Search Algorithm

```
1: Compute the errors for the GP and long-range function and min($\mathbf{r}$) at each
   configuration in the reference set.
2: Order the configurations from smallest to largest in terms of min($\mathbf{r}$).
3: Approximate $\mathrm{SSE}_{\mathrm{tot}}$ initially from the squared errors of the long-range func-
   tion alone.
4: Iterate through the min($\mathbf{r}$) from the lowest to the highest:
     • set $R_{\mathrm{cross}} = \min(\mathbf{r})$, which moves a single configuration from $\mathcal{A}_{\mathrm{LR}}$ to
       $\mathcal{A}_{\mathrm{GP}}$;
     • update $\mathrm{SSE}_{\mathrm{tot}}$ by deducting the squared long range error of the moved
       point from $\mathrm{SSE}_{\mathrm{LR}}$ and adding its squared GP error to $\mathrm{SSE}_{\mathrm{GP}}$;
     • store the new $\mathrm{SSE}_{\mathrm{tot}}$.
5: Find the smallest value of $\mathrm{SSE}_{\mathrm{tot}}$ and the value of $R_{\mathrm{cross}}$ to which it corre-
   sponds.
```

calculated once at the start. The GP predictions need to be re-calculated whenever the GP is updated, which occurs once per stage of training. However, these predictions are already required to choose the new training point at each training stage. Furthermore, calculation of all possible squared error values (step 4) is cheap because calculating each value in order requires only a simple update of $\mathrm{SSE}_{\mathrm{LR}}$ and $\mathrm{SSE}_{\mathrm{GP}}$. Consequently, a direct search is fast compared to the other steps of the sequential design algorithm and can therefore be undertaken at each design step with negligible additional computational effort.

3.5 Orthogonal direct search

An adaptation of the direct search algorithm is necessary for $\mathrm{C}_{\mathrm{multi}}$, as this requires a multidimensional optimisation of multiple classifier parameters. This is called the orthogonal direct search as it optimises one element of $\mathbf{R}_{\mathrm{cross}}$ while keeping the others fixed. The single element that varies is optimised using the direct search algorithm, as this is guaranteed to return the best minimum along that 1D slice of $\mathbf{R}_{\mathrm{cross}}$. Although orthogonal optimisations can be time-consuming, the speed of the one-dimensional direct search means that repeating it multiple

Algorithm 2 Orthogonal Direct Search Algorithm

```
1: Choose limits $NI_{\max}$ and $NR_{\max}$ on the number of iterations and restarts
   respectively (see the text for details).
2: Assemble the array of minimum distances $\overline{\mathbf{M}}$, as follows:
     • For every configuration, collect all interatomic distances which comprise
       the same atoms types into group $d$ and find $\min_d(\mathbf{r})$ for each $d$.
     • Arrange the lists of $\min_d(\mathbf{r})$ values in an $N$ x $D$ array, $\overline{\mathbf{M}}$, where $N$ is
       the number of configurations in the data set and $D$ is the number of
       unique atomic pairs in $\mathbf{r}$.
     • For each column in $\overline{\mathbf{M}}$, order the values from smallest to largest to
       produce an ordered list in each column.
3: For each restart, select a row from $\overline{\mathbf{M}}$ at random to be the initial guess at
   $\mathbf{R}_{\text{cross}}$.
4: For each $d$ in turn, fix all cross-over distances apart from $R_d$ and find the
   optimal value of $R_d$ using a direct search.
5: Repeat step 4 until $NI = NI_{\max}$ or the elements of $\mathbf{R}_{\text{cross}}$ remain unchanged;
   save this $\mathbf{R}_{\text{cross}}$ and its corresponding $\text{SSE}_{\text{tot}}$.
6: Repeat steps 3-5 until $NR = NR_{\max}$.
7: Select the $\mathbf{R}_{\text{cross}}$ that corresponds to the lowest value of $\text{SSE}_{\text{tot}}$.
```

times for all cross-over distances is feasible.

The orthogonal direct search proceeds via algorithm 2. As in the one-dimensional direct search, the square errors of the long-range and GP predictions at each configuration are pre-computed for this algorithm. The orthogonal search algorithm is not guaranteed to find the global minimum, as local minima may exist in the RMSE landscape. Hence, $NR_{\max}$ restarts are performed with randomly selected starting points. Each restart involves $NI_{\max}$ optimisations of each R_d. Here $NI_{\max} = 15$ and $NR_{\max} = 5$ for all systems. If the values in $\mathbf{R}_{\text{cross}}$ converge such that further one-dimensional searches in any orthogonal direction do not change its elements prior to $NI = NI_{\max}$, $\mathbf{R}_{\text{cross}}$ is saved and the next restart undertaken. In fact, it was rare that NI reached $NI_{\max}$ for any of the systems explored here. Moreover, despite the low $NR_{\max}$ employed, the same minimum was usually found across multiple restarts.

Algorithm 3 Constrained Placement Algorithm

```
1: Select the configuration in the reference set with the highest energy; add this
   configuration to the training set and remove it from the reference set.
2: Retrain the GP to the updated training set.
3: Determine the boundary that minimises the RMSE against the reference set
   via algorithm 1 or 2 as appropriate.
4: Find the configuration in $\mathcal{A}_{\text{GP}}$ for which the GP error is highest, where $\mathcal{A}_{\text{GP}}$
   is defined by the boundary from the previous step.
5: Add this configuration to the training set and remove it from the reference
   set.
6: Repeat steps 2-5 until the desired $\text{RMSE}_{\text{test}}$ or number of training points is
   reached.
```

The direct search offers a method that is at once fast, designed specifically for the discrete-stepped surface and perfect in a single dimension. This latter property means it skips over any local minima in a given orthogonal direction. As such, optimal cross-over distances for C_{multi} are obtained quickly and reproducibly under the orthogonal direct search. Typically, about 80 % of random restarts return the same set of cross-over distances for a given system and training set.

3.6 Training point placement methods

In addition to a classifier, any sequential design training strategy requires a method of choosing training points. Once the classifier parameters are optimized, the potential is specified and the next training point is determined from the highest error method. The point with the greatest error can be selected either from A_{GP} alone or from the union of $\mathcal{A}_{\text{GP}}$ and $\mathcal{A}_{\text{LR}}$. Using $\mathcal{A}_{\text{GP}}$ alone is referred to as the constrained placement method, which proceeds via algorithm 3. Steps 2-5 of this algorithm comprise a stage of training, with the $\text{RMSE}_{\text{test}}$ also calculated at each such stage.

Choosing new points from the union of $\mathcal{A}_{\text{GP}}$ and $\mathcal{A}_{\text{LR}}$ is named the open placement method. This proceeded identically to the constrained placement method (algorithm 3) except the highest error point was from either $\mathcal{A}_{\text{GP}}$ or $\mathcal{A}_{\text{LR}}$. In

$\mathcal{A}_{\mathrm{LR}}$ the highest error was found from the predictions of the long-range-function, whilst in $\mathcal{A}_{\mathrm{GP}}$ the GP predictions were used.

Models were trained under both point placement methods because each held potential advantages over the other. The constrained placement method ensures all training points are added in the GP region and so are of immediate use in prediction. Meanwhile, the open placement method is capable of immediately placing points in regions of the PES where the long-range function performed poorly, potentially transferring these regions to $\mathcal{A}_{\mathrm{GP}}$ more rapidly than under the constrained placement approach.

3.7 Closest model training strategy

A further training strategy, which is intended for comparison only, is the closest model training strategy. This employs the open placement method to select training points and classifies a configuration using $\mathrm{C}_{\mathrm{optimal}}$, where

$$\mathrm{C}_{\mathrm{optimal}}(\mathbf{r}) = \begin{cases} \mathcal{A}_{\mathrm{GP}}, & \text{if } \mathrm{SE}_{\mathrm{GP}} \leq \mathrm{SE}_{\mathrm{LR}} \\ \mathcal{A}_{\mathrm{LR}}, & \text{otherwise} \end{cases} . \tag{3.8}$$

Here, $\mathrm{SE}_{\mathrm{method}}$ is the squared error in the prediction from“method” at $\mathbf{r}$. $\mathrm{C}_{\mathrm{optimal}}$ is so named because it classifies a configuration based on whether GP regression or the long-range function best approximate its energy.

Equation (3.8) shows that $\mathrm{C}_{\mathrm{optimal}}$ employs no boundary and so requires prior knowledge of the energy of a configuration in order to classify it. Consequently, models obtained from the closest model strategy are unsuitable for prediction. However, this method represents an ‘ideal’ classifier that is guaranteed to find the optimal $\mathrm{RMSE}_{\mathrm{test}}$ for a given GP model. Hence $\mathrm{C}_{\mathrm{optimal}}$ is useful for estimating how inaccuracies in the parametric classifiers affect training efficiency. If models from $\mathrm{C}_{\mathrm{optimal}}$ significantly outperform the other classifiers, the other classifiers are too simple to properly approximate the true boundary.

There are circumstances where $\mathrm{C}_{\mathrm{optimal}}$ may not result in optimal GP training when assessed via the RMSE against the test set. Short-range hypersurfaces where the interaction energy is predicted near-exactly by the long-range function

Table 3.3: The classifier and training point placement method for all training strategies examined in this chapter.

Training Strategy	Classifier	Point Placement Method
Single-Constrained	$\mathrm{C}_{\mathrm{single}}$	Constrained Placement
Multi-Constrained	$\mathrm{C}_{\mathrm{multi}}$	Constrained Placement
Single-Open	$\mathrm{C}_{\mathrm{single}}$	Open Placement
Multi-Open	$\mathrm{C}_{\mathrm{multi}}$	Open Placement
Closest Model	$\mathrm{C}_{\mathrm{optimal}}$	Open Placement
Fixed Boundary	$\mathrm{C}_{\mathrm{fixed}}$	Constrained Placement

due to chance may exist. Points in the reference set that are near these hypersurfaces will be classified by $\mathrm{C}_{\mathrm{optimal}}$ as part of $\mathcal{A}_{\mathrm{LR}}$ instead of $\mathcal{A}_{\mathrm{GP}}$. If the configuration density of the reference set around such a hypersurface is insufficiently high, no training points will be added in its vicinity. Consequently, points in the test set near to the hypersurface will be inadequately approximated by either method of prediction; the GP will perform poorly due to a lack of nearby training points and the long-range function will be inaccurate for short-range points that are not extremely close to the hypersurface. This problem was avoided by using dense reference sets and by not using the constrained placement strategy with $\mathrm{C}_{\mathrm{optimal}}$.

3.8 Overview of training strategies

As stated, the sequential design method, or training strategy, requires a choice of classifier and placement strategy. The combinations of these examined in this chapter are given in table 3.3. Methods that do not involve $\mathrm{C}_{\mathrm{optimal}}$ can make predictions and so are suitable for applications. The method involving $\mathrm{C}_{\mathrm{optimal}}$ can only classify points if the true energy is already known, and so is only useful to estimate the loss in performance due to inaccuracies in the parametric classifiers. The fixed boundary method corresponds to the method of Uteva *et al.* [29] and is included to allow comparison with this prior method, on which the new methods build. The method of Uteva *et al.* [29] was previously shown to significantly reduce the number of training points compared to LHC design. It is demonstrated

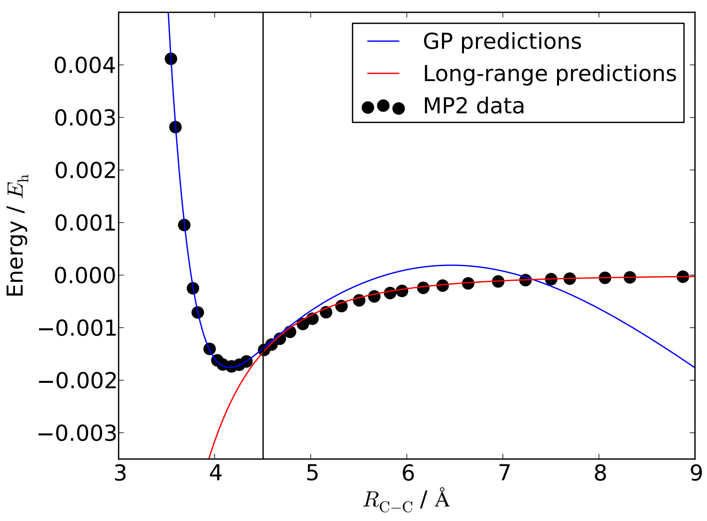


Figure 3.2: The predictions of the GP (blue) and the long-range function (red) for a slice through the PES of the $(CO_2)_2$ potential in which the two molecules are in a T-shaped configuration. The cross-over distance for this model is shown by the black line at 4.505 Å. Data points (black circles) are from MP2 calculations that are independent of the GP training data.

below that these new boundary optimisation methods improve further the already efficient methods of Uteva *et al.* [29].

3.9 Results for non-$(HX)_2$ potentials

Comparisons of the performances of the different training strategies are made using the HF-Ne, HF-Na^+, CO-Ne, CO_2-Ne, $(CO_2)_2$, $(HF)_2$ and $(HCl)_2$ potentials. These were selected as they provide a range of interaction types and well depths to test the robustness of the new training strategies. However, poor performance of fixed boundary training on the latter two potentials means that these are discussed separately in section 3.10.

The number of training points is N_t, which in each system was less than 10 % of the number of configurations in the corresponding reference set (see table 3.1 and the related text). Consequently, the training sets for the models discussed are candidates for transfer learning because their small size makes, for example, CCSD(T) calculations possible for the whole training set.

Boundary optimisation is anticipated to improve training efficiency as, for

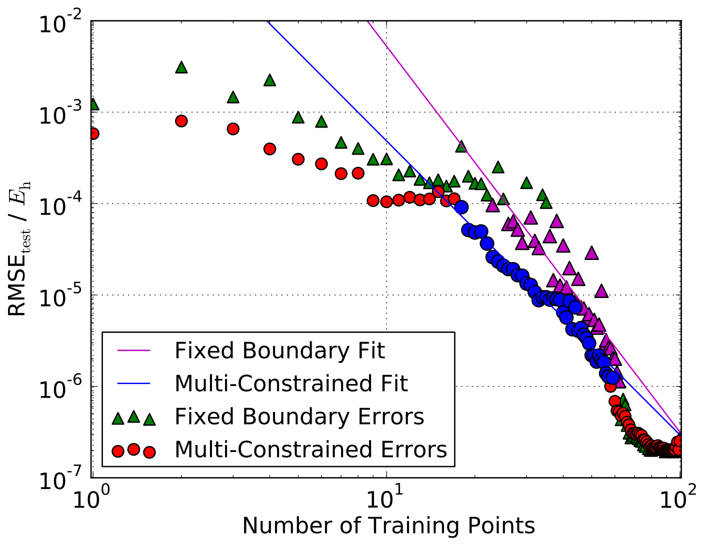


Figure 3.3: Plot of $\text{RMSE}_{\text{test}}$ / E_{h} against N_{t} on a $\log_{10}$ scale for models of the HF-Ne PES trained via the multi-constrained (circles) and fixed boundary (triangles) training strategies. The blue circles and pink triangles are points which were included in the fitting of the lines shown.

small N_{t}, the long-range function will outperform the GP at the outer edge of the potential well. This is illustrated in figure 3.2 for the $(CO_2)_2$ system, using a GP model trained under the single-constrained strategy up to $N_{\text{t}} = 21$. This figure shows that the predictive accuracy of the long-range function exceeds that of GP regression for configurations with C-C separations above the cross-over distance. Despite this, in the fixed boundary method these configurations would be predicted with the GP. This not only reduces overall accuracy, but also means new training points must be placed at long range to address this, when allowing the long-range function to predict these energies would give adequate accuracy.

The impact of the above on training efficiency compared to fixed boundary training is shown for HF-Ne in figure 3.3. For C_{multi} the RMSE falls faster with N_{t} compared to C_{fixed}, indicating improved training efficiency. This improvement is most pronounced when N_{t} is low and so $\text{RMSE}_{\text{test}}$ is high. This is expected because increasing the size of the training set increases the size of the GP region, meaning R_{cross} in the boundary-optimised strategies will approach the fixed value of 8.5 Å. Consequently, the difference between the fixed boundary and boundary-optimised models will close as N_{t} increases. Equivalent plots to figure 3.3 for all training strategies for all systems are found in Appendix E.

Figure 3.3 also shows that the RMSE data are somewhat noisy. Possible causes of this noise are minor variations in the hyperparameters upon retraining, the discrete nature and stochastic design of the reference set, and the low values of N_{t} meaning that addition of a single point has a non-smooth effect on the RMSE. The new methods herein have considerably lower noise than the prior fixed boundary method.

To compare the efficiency gain over fixed boundary training across all new strategies from subsections 3.7 and 3.8, a metric, E, is used. This compares the N_{t} required by two different strategies to achieve a given $\mathrm{RMSE}_{\mathrm{test}}$. When comparing a new training strategy with fixed boundary training,

$$E(\mathrm{RMSE}_{\mathrm{test}}) = \frac{N_{\mathrm{t}}(\mathrm{RMSE}_{\mathrm{test}})}{N_{\mathrm{t,fixed}}(\mathrm{RMSE}_{\mathrm{test}})}\mathrm{x}100\%, \tag{3.9}$$

where N_{t} and $N_{\mathrm{t,fixed}}$ are the numbers of training points required by the new strategy and fixed boundary training, respectively. These quantities and E are shown as functions of $\mathrm{RMSE}_{\mathrm{test}}$ as they vary with its value.

For a given $\mathrm{RMSE}_{\mathrm{test}}$, the values of N_{t} and $N_{\mathrm{t,fixed}}$ are determined from least squares fits of $\log_{10}(\mathrm{RMSE}_{\mathrm{test}})$ versus $\log_{10}(N_{\mathrm{t}})$, with examples of such fits shown in figure 3.3 for the HF-Ne potential. Fits are made in the region where the RMSE decays as a power law of N_{t}. For all systems this corresponds to $1 \text{ x } 10^{-6}\ E_{\mathrm{h}} \leq \mathrm{RMSE}_{\mathrm{test}} \leq 1 \text{ x } 10^{-4}\ E_{\mathrm{h}}$, apart from HF-Na^{+} where the region is $1 \text{ x } 10^{-5}\ E_{\mathrm{h}} \leq \mathrm{RMSE}_{\mathrm{test}} \leq 1 \text{ x } 10^{-3}\ E_{\mathrm{h}}$ due to the larger high energy cut-off for this potential. The fits provide continuous lines to interpolate to any $\mathrm{RMSE}_{\mathrm{test}}$ within the stated ranges. This enables a comparison of N_{t} between training strategies at fixed $\mathrm{RMSE}_{\mathrm{test}}$ that accounts for the somewhat noisy data. Moreover, this is the range of errors in which the models are useful for applications and the decrease in $\log(\mathrm{RMSE}_{\mathrm{test}})$ with $\log(N_{\mathrm{t}})$ is linear.

The fitted equations for each training strategy for the CO-Ne system are given in table 3.4 as power laws in N_{t}. Equivalent tables for all other systems are found in Appendix E. As the data are noisy, these tables also show the R^2 value of each fit. In the case of CO-Ne, these evidence the high quality of the fits. In fact, no fit from any training strategy for any system achieves an R^2 value lower than 0.8907 (from the fixed boundary training strategy on the HF-Na^{+} system).

Table 3.4: The equation of the lines of best fit for $\text{RMSE}_{\text{test}}$ of models from all training strategies for CO-Ne as power laws in the number of training points, N_{TP}. Also shown are the R^2 values of each fit on the data. All fits were over points in the range 1 x 10^{-6} $E_{\text{h}} \leq \text{RMSE}_{\text{test}} \leq$ 1 x 10^{-4} E_{h}.

Training Strategy	Line of best fit to $\text{RMSE}_{\text{test}}$	R^2
Single-Constrained Placement	$4.694N_{\text{TP}}^{-3.842}$	0.929
Multi-Constrained Placement	$1.551N_{\text{TP}}^{-3.647}$	0.944
Single-Open Placement	$2.901N_{\text{TP}}^{-3.718}$	0.921
Multi-Open Placement	$3.113N_{\text{TP}}^{-3.830}$	0.958
Closest Model	$0.865N_{\text{TP}}^{-3.490}$	0.944
Fixed Boundary	$546.0N_{\text{TP}}^{-5.069}$	0.914

Furthermore, figure 3.3 and the corresponding plots in Appendix E show that the RMSE data follow a straight line (in a log-log plot). This implies that the R^2 arises from scatter in the data rather than unsuitability of the fitting function.

Re-arranging the equations in table 3.4 provides expressions for N_{t} in terms of $\text{RMSE}_{\text{test}}$ for the CO-Ne potential. These give E as a function of $\text{RMSE}_{\text{test}}$, via equation (3.9), for all new training strategies herein. This was done for all other potentials as well, with plots of E against $\text{RMSE}_{\text{test}}$ for all training strategies for each system given in figure 3.4.

As observed earlier, the training efficiency gains in figure 3.4 are more pronounced at high $\text{RMSE}_{\text{test}}$ for all training strategies across all systems. This suggests that boundary optimisation is most effective when the training set is small, making this technique ideal for applications where a computationally cheap but less accurate potential is required. Nevertheless, significant reductions in the number of training points are also obtained in the RMSE range where PESs become useful for first principles predictions.

For example, a GP potential with an RMSE of 3 x 10^{-4} eV per atom (1.1 x 10^{-5} E_{h} per atom) was employed in a recent simulation of the thermal properties of β-Ga_2O_3 [42]. Furthermore, Uteva *et al.* successfully determined the CO_2-CO second virial coefficient using a GP PES with an RMSE of 2.4 x 10^{-5} E_{h} [27]. In this RMSE range the boundary optimisation methods typically reduce the required number of training points by 15-33% (see figure 3.4).

Generally, the closest model strategy generates the largest efficiency gain, while the smallest improvement comes from strategies involving C_{single}. Efficiency gains only slightly below those from the closest model strategy are achieved by the strategies that use C_{multi}. This hierarchy of improvement implies that the choice of classifier, rather than point placement strategy, is most important because strategies with the same classifier perform more similarly than those with the same point placement method. The closest model strategy is included only to illustrate the total training efficiency gain possible from an 'ideal' classifier, and it is encouraging that the best boundary optimisation methods are close to this ideal case. Indeed, the difference in E between this ideal method and the closest-performing C_{multi} strategy never exceeds ~ 3 % for any system. This implies that C_{multi} captures the true shape of the boundary region for the systems explored sufficiently. Thus introducing a more detailed classifier would not be worth the increased cost of developing and evaluating the potential.

C_{multi} always outperforms C_{single}. However, for the CO_2-Ne potential (figure 3.4d), C_{multi} also outperforms the closest model strategy. This is because the long-range function is nearly exact for a group of short-range configurations in the reference set. This is shown in figure 3.5 by the thin 'peninsula' of points that are best estimated by the long range function (in red) which encroaches deep into the GP region (in blue). This 'peninsula' exists because the long-range function is of higher energy than the MP2 data in some regions of the PES and of lower energy in others, meaning there must be some hypersurface in between where the two are equal. Prediction for test configurations close to the 'peninsula' is problematic under C_{optimal} unless the reference set is very dense in this region. This is because the near-exact predictions of the long-range function on the 'peninsula' mean no training points are added there, leading the closest model strategy to perform relatively poorly for the CO_2-Ne potential.

The total gain in training efficiency achieved by boundary optimisation varies somewhat between systems. While the best-performing training strategy for HF-Ne improved training efficiency by between 25-39 % (*i.e.* E = 61-75 %), for $(CO_2)_2$ the gain was only 12-18 % (E = 82-88 %). The more limited gains for

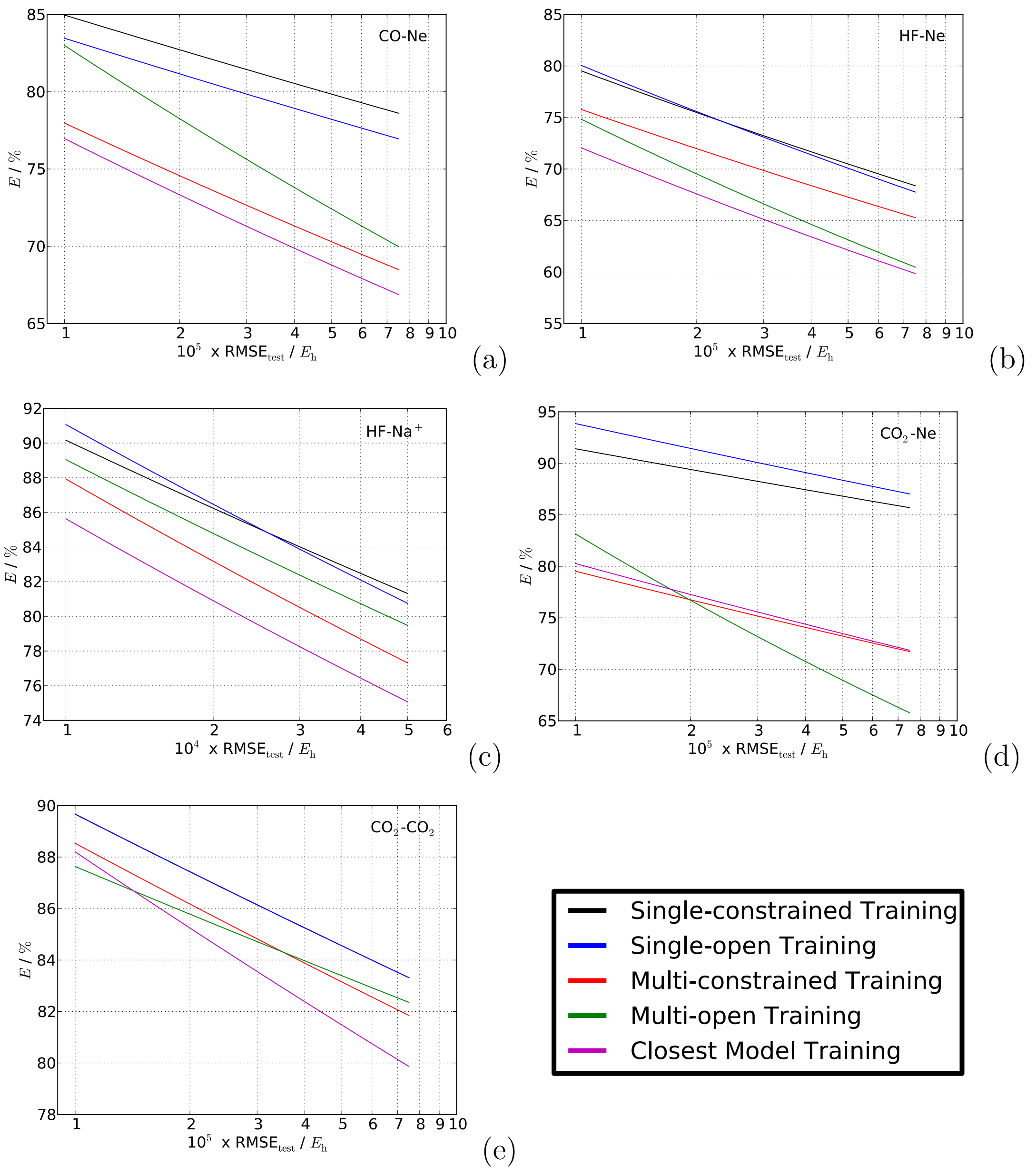


Figure 3.4: Plots of E against RMSE$_{\mathrm{test}}$ for all potentials are shown in parts (a) to (e). The potential referred to in each frame is shown in the upper right corner.

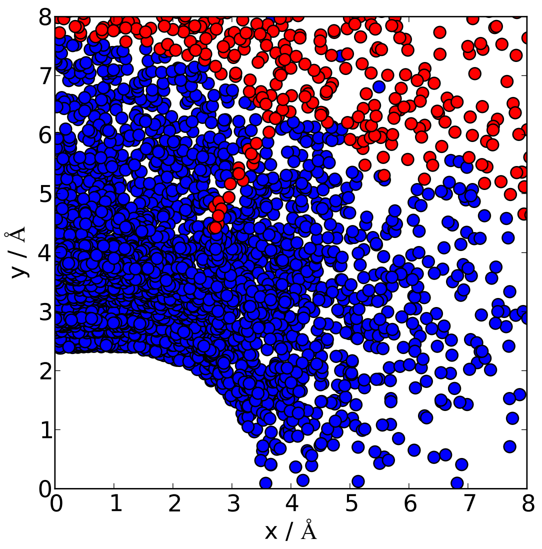


Figure 3.5: Plot of the x and y coordinates of the Ne in the CO_2-Ne system for configurations in the reference set. The CO_2 molecule is aligned along the x-axis with the C at the origin. Points are classified as GP points (blue) or asymptotic points (red) by C_{optimal} using a potential from the closest model strategy trained up to $N_t = 100$.

$(CO_2)_2$ may be because the *a priori* choice of $R_{\text{cross}} = 8.5$ Å, required for the fixed boundary method, is reasonable for this system. Nevertheless, even for the $(CO_2)_2$ potential, use of multi-constrained training confers an efficiency gain of ~ 18 % over fixed boundary training. This means that for all of the potentials explored, use of a training strategy that employs C_{multi} confers a useful improvement over fixed boundary training.

Boundary optimisation improves the training efficiency due to more effective placement of training points. This is illustrated in figure 3.6, which shows, for HF-Ne, the differences in training point placement for three training methods: fixed boundary, single-constrained and closest model. While all place most points at separations below 3 Å, indicating that the repulsive wall is the hardest region to model, the placement of points at larger separations diverges between methods.

For the first 20 training points few configurations are placed beyond 3 Å for the single-constrained and closest model strategies. Thereafter, the distance at which training points are added slowly increases, even for the closest model strategy, which does not employ a boundary directly. In fact, single-constrained training

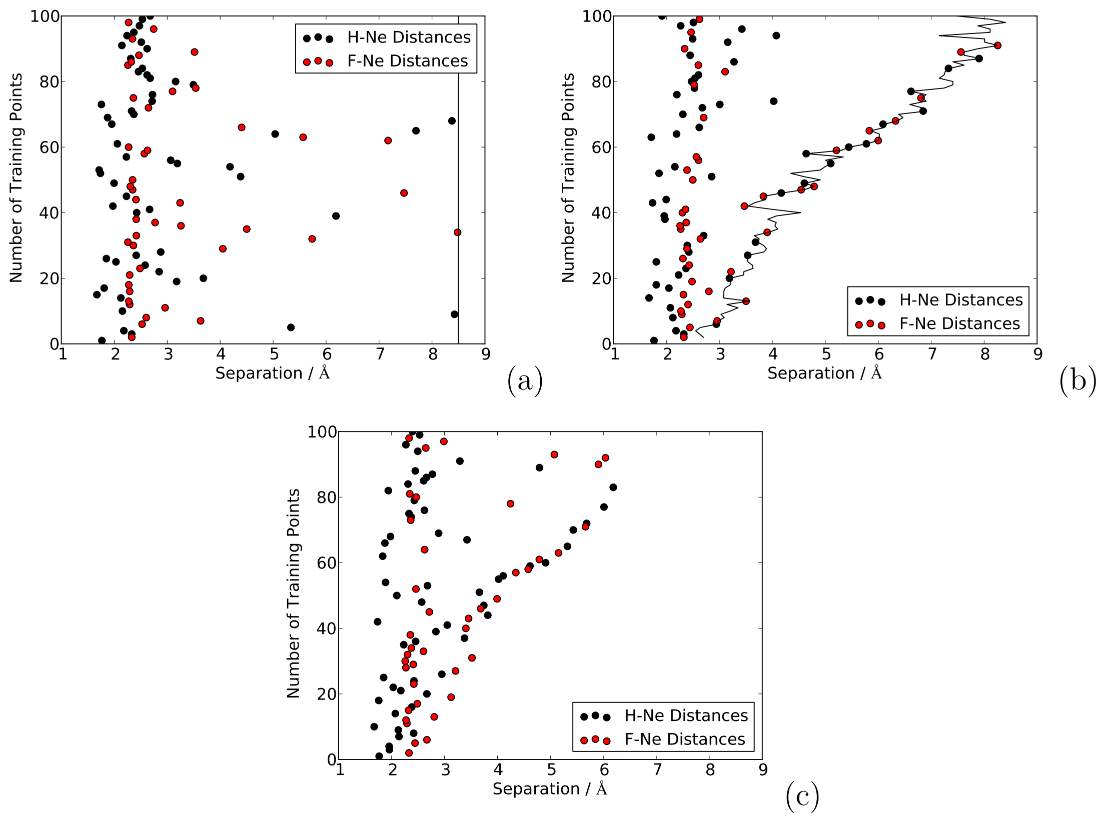


Figure 3.6: Plots showing training point placement for the first 100 training points for models of the HF-Ne PES trained using the fixed boundary (a), single-constrained placement (b) and closest model (c) strategies. These points are coloured based on the shortest interatomic distance in the configuration, with only this shortest distance shown for each. Boundary values are represented by black lines where applicable.

adds training points beyond 8 Å only after $\sim$ 90 training points have been placed and the closest model strategy does not add any training points above 7 Å at all.

In contrast, the fixed boundary training method adds its fifth training point at a minimum separation above 5 Å and its eighth at 8.5 Å. This demonstrates that a fixed boundary strategy switches between placing training points in the repulsive wall and at the boundary from the onset of training. This difference is because fixed boundary training requires the GP to predict energies at separations up to 8.5 Å from the start of training. Consequently, training points must be added at separations near the 8.5 Å boundary from the onset, even though the energies at these separations are very small and generally well-approximated by the long-range function.

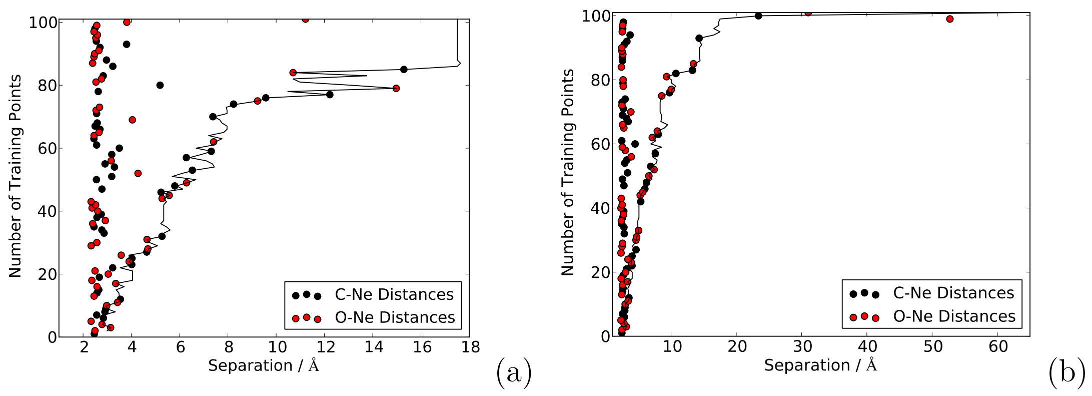


Figure 3.7: Plots showing training point placement for the first 100 training points for models of the CO-Ne PES trained using the single-constrained placement (a) and single-open placement (b) strategies. These points are coloured based on the shortest interatomic distance in the configuration, with only this shortest distance shown for each. Boundary values are represented by black lines.

This contrasts with the boundary-optimised and closest model strategies, which allow the long-range function to approximate configurations at separations around 8.5 Å when the number of training points is low. This facilitates more efficient model development under boundary-optimised and closest model training because point placement can be focused on the short-range region of the PES, where energy varies more rapidly with configuration.

Training point plots for the CO-Ne potential, given in figure 3.7, show that the single-open strategy extends $\mathcal{A}_{\mathrm{GP}}$ much further than single-constrained training. This suggests that the capacity to place points in $\mathcal{A}_{\mathrm{LR}}$ facilitates faster expansion of the boundary for this system. However, these plots also indicate that this discrepancy is most noticeable when the number of training points is large; specifically, rapid expansion of the boundary under single-open training was instigated by placement of a single training point at long range, which facilitated the increase in R_{cross} to 63.0 Å. Prior to this, the cross-over value from single-open training was quite similar to that from single-constrained training (both were $\sim$ 17 Å). Hence the two training strategies differ significantly only when the predictions from GP regression are already highly accurate, as at this stage the predictions of the long-range function are poor enough by comparison to merit placement of a training point at long-range.

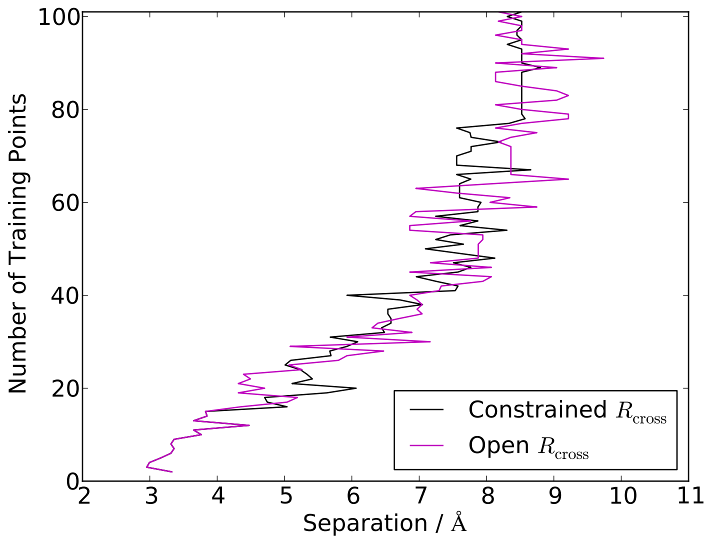


Figure 3.8: Plot showing the values of R_{cross} achieved for the first 100 training points by single-open and single-constrained training for models of the CO_2-Ne potential.

The other systems also show close similarity between the cross-over distances for the single-constrained and single-open strategies, as illustrated in figure 3.8. Equivalent plots for other systems are found in Appendix E. Figure 3.8 shows that for the CO_2-Ne potential the values of R_{cross} from the two strategies are similar throughout training. Such a trend suggests that the choice of point placement strategy does not significantly change the value of R_{cross}, implying once more that the choice of classifier is of greater import.

Similarities in the evolution of the boundary are seen when the cross-over distances achieved under C_{multi} and C_{single} are compared for a given point placement method. Figure 3.9 shows the results of such a comparison for the HF-Na^+ and CO_2-Ne systems. The cross-over distances generally grow at a similar rate, but the value of R_{cross} is consistently larger than all values in $\mathbf{R}_{\text{cross}}$. This suggests that there are differences between C_{single} and C_{multi}, which manifest in both the training efficiency and boundary placement.

The larger GP region under C_{single} compared with C_{multi} is explained by noting that both are approximations of an 'ideal' classifier. When N_{t} is low (*i.e.* one or two), such a classifier will attempt to make $\mathcal{A}_{\text{GP}}$ as small as possible because the training set comprises configurations from the repulsive wall only. Consequently,

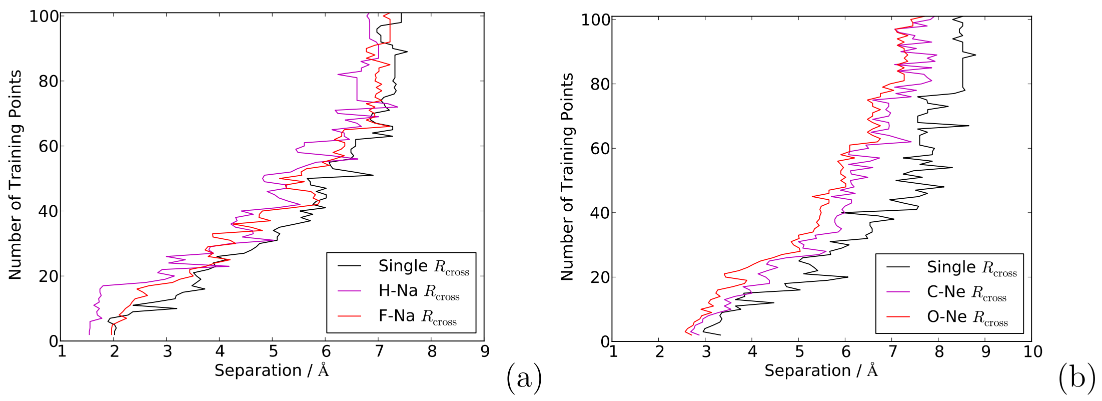
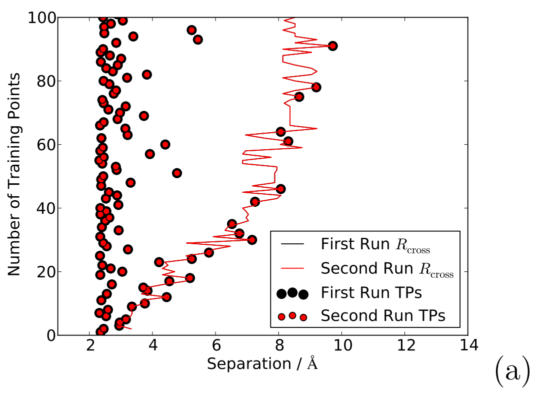


Figure 3.9: Plots showing the value of R_{cross} and entries in $\mathbf{R}_{\text{cross}}$ for the first 100 training points for models of the HF-Na$^+$ potential from single/multi-open training (a) and the CO_2-Ne potential from single/multi-constrained training (b).

given the same training set, C_{single} and C_{multi} also try to minimise the size of $\mathcal{A}_{\text{GP}}$. As C_{multi} is more flexible its approximation of the 'ideal' classifier will be closer than that of C_{single}, meaning $\mathcal{A}_{\text{GP}}$ under C_{single} will be larger. Other than the repulsive wall, the largest errors tend to occur at separations around the boundary. Thus, due to its larger $\mathcal{A}_{\text{GP}}$, a C_{single} strategy will place its first long-range training point at a larger separation than a C_{multi} strategy. This facilitates faster expansion of the boundary under C_{single} than C_{multi} regardless of which point placement method is used.

Figure 3.9a shows that $R_{\text{H-Na}^+} < R_{\text{F-Na}^+}$ throughout most of training for the HF-Na$^+$ potential, meaning that the interaction involving the larger atom (F) obeys a larger cross-over distance. Such ordering of the cross-over distances also applies to the HF-Ne system for most of training. Moreover, figure 3.9b shows that the ordering of the cross-over distances for the CO_2-Ne potential is consistent, with $R_{\text{O-Ne}} < R_{\text{C-Ne}}$ throughout training. In fact, for the $(CO_2)_2$ and CO-Ne potentials it holds generally throughout training that $R_{\text{O-O}} < R_{\text{O-C}} < R_{\text{C-C}}$ and $R_{\text{O-Ne}} < R_{\text{C-Ne}}$ respectively for both point placement methods. This implies that the ordering of the cross-over distances under C_{multi} is consistent between systems as well as between atomic sizes.

Additionally, figures 3.6-3.9 show that the cross-over distance increases with the size of the training set. This is expected because a larger training set means the GP has more information with which to infer the function describing the PES.



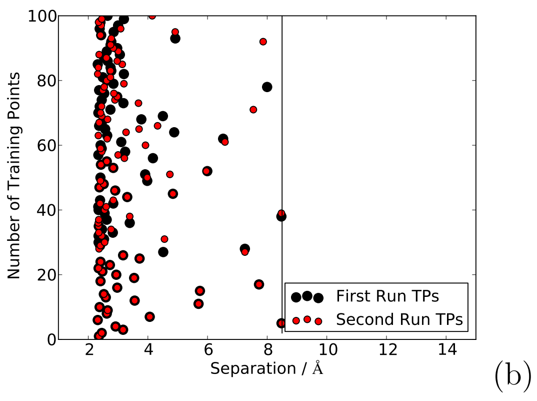


Figure 3.10: Plots showing the first 100 training points and values of R_{cross} from two runs of single-open training (a) and fixed boundary training (b) for models of the CO_2-Ne potential.

The resultant, more accurate GP enables larger cross-over distance(s). This is the case for all training strategies across all systems.

Boundary optimisation methods decrease the cost of using PES, by reducing the number of training points and the size of the GP region. Fewer training points means GP evaluations are cheaper, as the GP cost is proportional to N_{t}. Additionally, the lower R_{cross} or $\mathbf{R}_{\text{cross}}$ of the boundary-optimised model means that in any application the GP would be used less often, in favour of the much cheaper asymptotic function. Hence boundary optimisation produces models that are more efficient to implement in applications than fixed boundary training, with no reduction in overall accuracy.

Though R_{cross} or $\mathbf{R}_{\text{cross}}$ in excess of 8.5 Å are seen in some plots in Appendix E, these coincide with potentials trained on large N_{t}. They are therefore in the plateau, seen after $N_{\text{t}} \approx 70$ in figure 3.3, where increasing the size of the training set has little associated benefit to model accuracy. Thus, the GP is only marginally more accurate than the long-range function at the configurations added to $\mathcal{A}_{\text{GP}}$ at this stage and a similarly accurate model with a smaller boundary and training set could be chosen instead with little impact on predictive accuracy.

Boundary optimisation also increases the reproducibility of training. That is, for two separate models from the same training strategy and reference set, the results will be more similar for a boundary-optimised strategy compared to using

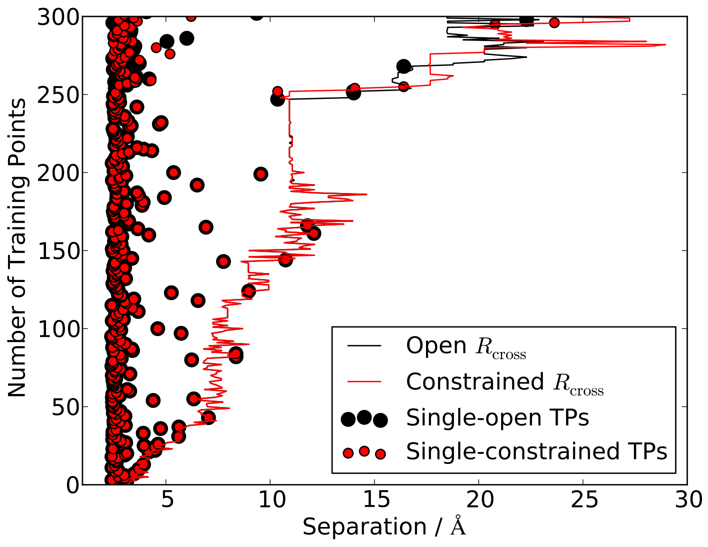


Figure 3.11: The first 300 training points and values of R_{cross} from single-open and single-constrained training for the $(CO_2)_2$ potential.

a fixed boundary. For example, when modelling the CO_2-Ne potential with single-open training the results from two separate training runs were identical, as shown in figure 3.10. This is noteworthy because the values of the hyperparameters selected when maximising $\log(\mathcal{L})$ (equation (2.9)) can vary slightly, even for the same training set. Such variations can alter the predictions of the GP, leading to different values for R_{cross} or $\mathbf{R}_{\text{cross}}$ and the selection of different training points. Thus, that two separate runs of the same training strategy are totally identical is encouraging.

Furthermore, figure 3.10 shows that for fixed boundary training separate runs were identical only up to $N_{\text{t}} = 27$. While the exact reproducibility in figure 3.10a is not present for the other potentials, there is generally significantly less difference between independent runs of the boundary-optimised training methods than for fixed boundary training. For example, models of the HF-Ne potential from single-constrained and single-open training are reproducible up to $N_{\text{t}} = 30$ and $N_{\text{t}} = 41$ respectively, compared with $N_{\text{t}} = 16$ for fixed boundary training. Equivalent plots to figure 3.10 for this system under these strategies are given in the supplementary material.

This reproducibility increase compared to fixed boundary training does not transfer to the use of the C_{multi} strategies, likely because a direct search is not as reproducible in multiple dimensions as in a single dimension. However, from the R^2 values in table 3.4 it can be seen that use of either C_{multi} or C_{single} reduces the scatter in the $\text{RMSE}_{\text{test}}$ data relative to use of C_{fixed}, as evidenced by the larger R^2 values achieved when training with the former two. This trend is repeated across all other potentials examined here, which suggests that use of boundary optimisation leads to more stability in selection of the hyperparameters and hence more consistent predictions.

For the $(CO_2)_2$ system, reproducibility is seen not just for repeat runs of identical training strategies but between the training strategies that use C_{single}. This is illustrated in figure 3.11, which shows that the single-constrained and single-open training strategies choose identical training points until $N_{\text{t}} = 210$. Such an observation explains why the two methods have identical E in figure 3.4e. This is because when $N_{\text{t}} = 210$ the $\text{RMSE}_{\text{test}}$ values for single-open and single-constrained placement were 2.817 x 10^{-7} E_{h} and 2.784 x 10^{-7} E_{h} respectively. Consequently the error was too low at this N_{t} to be included in the fit, meaning the two strategies were identical over the $\text{RMSE}_{\text{test}}$ values used in fitting. That boundary optimisation mitigates instability in hyperparameter selection to the extent that the highest-dimensional potential examined exhibited reproducibility up to $N_{\text{t}} = 210$ represents a significant improvement over fixed boundary training.

Table 3.5: The equation of the lines of best fit of models from the multi-constrained and fixed boundary training strategies for the $(HX)_2$ systems as power laws in the number of training points, N_{t}. Also shown are the R^2 values of each fit on the data.

Training Strategy	Equation of Line	R^2
$(HF)_2$ Multi-Constrained Placement	$0.3903 N_{\text{t}}^{-1.6858}$	0.9681
$(HCl)_2$ Multi-Constrained Placement	$0.0105 N_{\text{t}}^{-1.0056}$	0.9001
$(HF)_2$ Fixed Boundary	$1.7733 N_{\text{t}}^{-1.8401}$	0.7914
$(HCl)_2$ Fixed Boundary	$0.0301 N_{\text{t}}^{-1.0188}$	0.5516

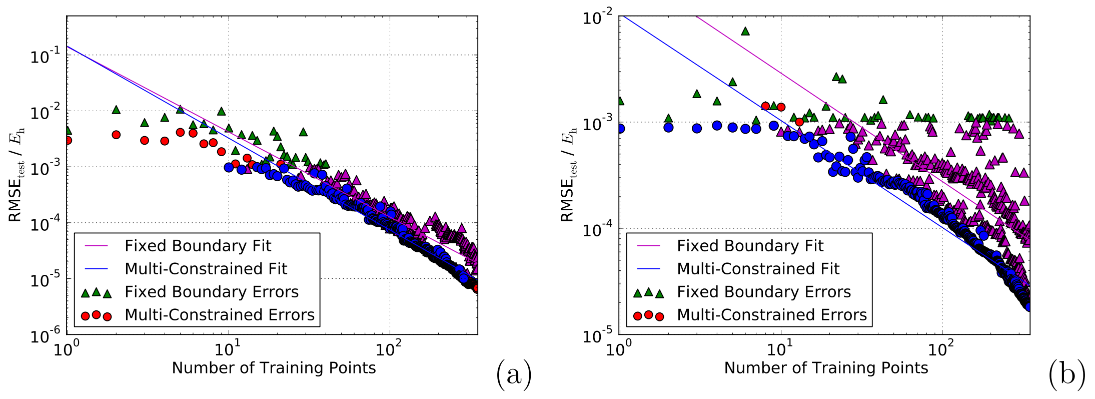


Figure 3.12: Plots showing the $\text{RMSE}_{\text{test}}$ values for the first 350 training points for models trained under the fixed boundary and multi-constrained strategies, as well as their associated fits for $(HF)_2$ (a) and $(HCl)_2$ (b). Points shown in blue (for multi-constrained) and purple (for fixed boundary) were used to develop the fits, the equations of which are in table 3.5.

3.10 Results for $(HX)_2$ potentials

The variability in hyperparameter selection that was alleviated by boundary optimisation was particularly problematic for the $(HF)_2$ and $(HCl)_2$ systems. In fact, the issue was such that comparison of boundary-optimised strategies with fixed boundary training via the efficiency metric E (equation (3.9)) was not feasible. This is illustrated in figure 3.12 and by the poor R^2 values associated with fixed boundary training in table 3.5.

Figure 3.12 shows that, for both dimers, fixed boundary training results in models for which the errors fluctuate markedly. This is most pronounced for $(HCl)_2$, where at any stage of training the predictive error of the fixed-boundary model appears to be in one of two regions. This implies heavily that the issue stems from the selection of hyperparameters, as each distinct set of errors in figure 3.12b will correspond to a set of hyperparameters.

Specifically, these results suggest that there are several sets of hyperparameters that correspond to local minima in $\log(\mathcal{L})$ and that all of these are similar in value. This is because the presence of a clear global minimum would lead to consistent hyperparameter selection between stages of training, promoting the consistent fall in $\text{RMSE}_{\text{test}}$ with N_t observed when using boundary optimisation.

Examination of the hyperparameters selected at each stage of training sup-

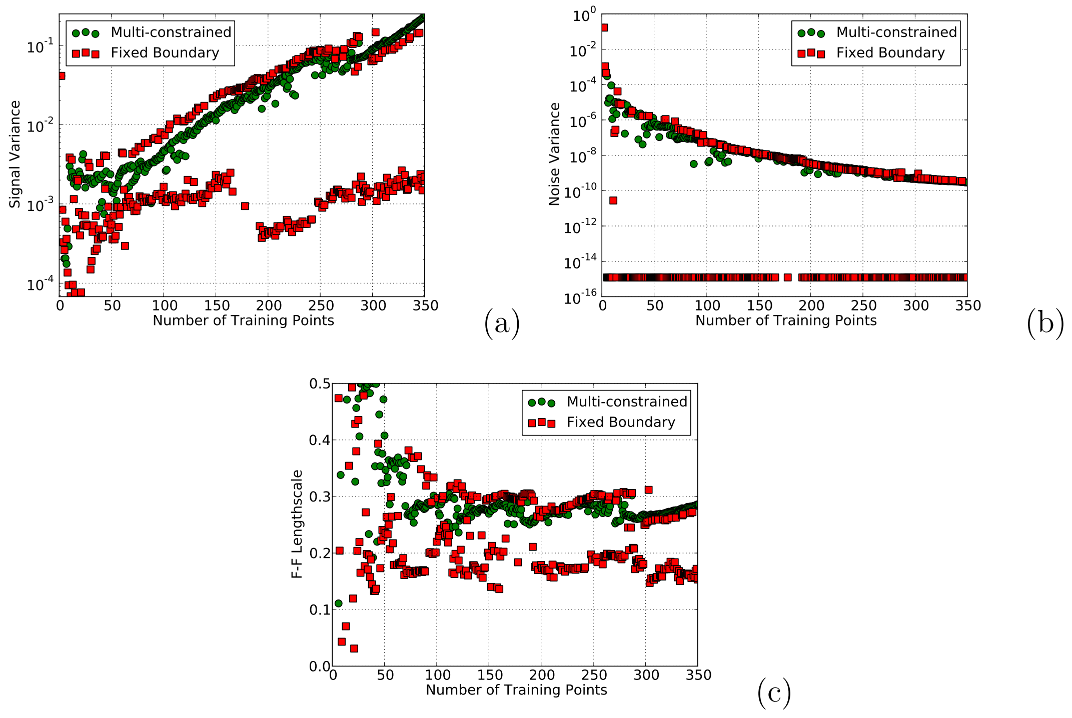


Figure 3.13: Plots showing the values of the signal variance (a), noise variance (b) and F-F lengthscale (c) for the first 350 training points for models of the $(HF)_2$ PES trained under the fixed boundary and multi-constrained strategies.

ports this theory. Figure 3.13 shows the signal variance, noise variance and F-F lengthscale (parts a, b and c respectively) selected by models from multi-constrained and fixed boundary training to the $(HF)_2$ PES. Equivalent plots for the $(HCl)_2$ potential are shown in figure 3.14.

The figures demonstrate that for all hyperparameters the values selected by fixed boundary training appear to be taken with a similar frequency from two regions. Meanwhile, the hyperparameters in the multi-constrained model seem to be taken from only one of these regions. Hence fixed boundary training produces models under which the hyperparameters, and therefore predictions, are inconsistent between stages of training. This contrasts with the boundary-optimised strategies, which produce models with more consistent hyperparameter values.

Two sets of hyperparameters are seen as equally plausible under fixed boundary training because the PES can generally be viewed as comprising two sections: the potential well and repulsive wall, and the long-range asymptotic region. In each section, the energy changes differently with configurational geometry, even

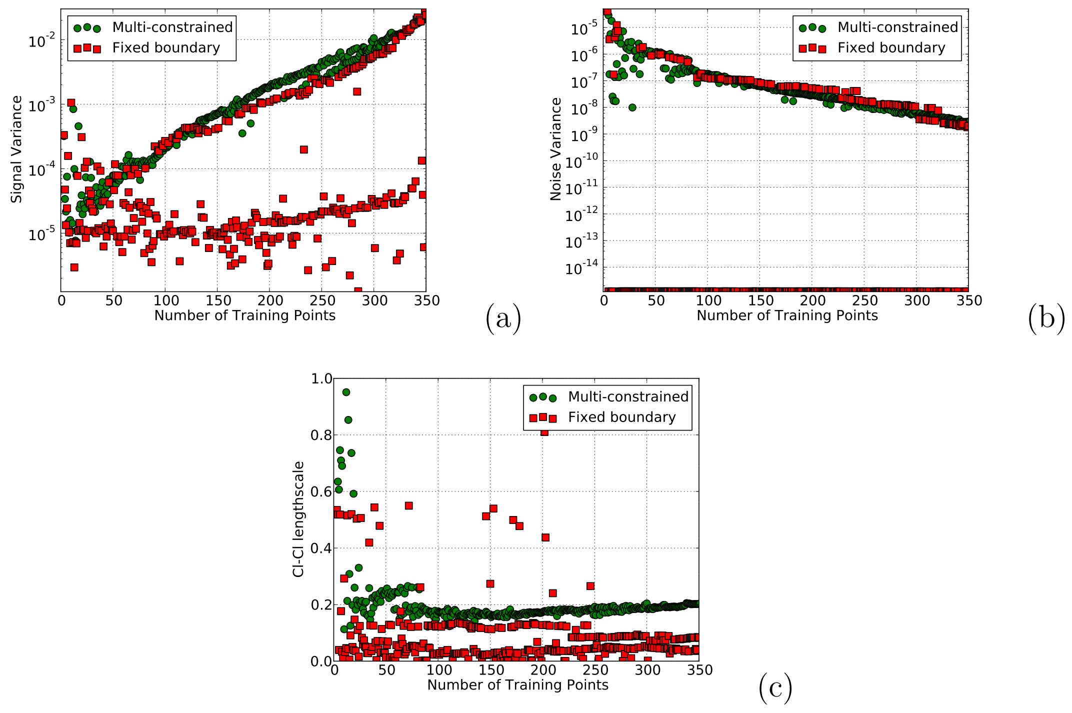


Figure 3.14: Plots showing the values of the signal variance (a), noise variance (b) and Cl-Cl lengthscale (c) for the first 350 training points for models of the $(HCl)_2$ PES trained under the fixed boundary and multi-constrained strategies.

under the $r \rightarrow r^{-1}$ transform.

Fixing $R_{\text{cross}} = 8.5$ Å means that, from the start of training, part of the long-range section of the PES will be included in the GP region. Thus, sections of the PES where different hyperparameters are required are modelled in concert by the GP throughout. This gives rise to the observed inconsistency in model accuracy under fixed boundary training.

This may be more noticeable for the $(HX)_2$ systems due to the dipole-dipole interactions, which were not present in the systems explored in section 3.9. These could exacerbate the issue of fluctuating variability of energy with separation as the X—X and H—H interactions are strongly repulsive, while the H—X interactions are strongly attractive. Thus the PES for each of these systems will be more variable than those in the last section.

However, the variability in hyperparameter selection is more pronounced for the $(HCl)_2$ dimer than the $(HF)_2$ dimer despite the dipole in the former being weaker. It is therefore likely that the disparity in atom sizes in these systems exac-

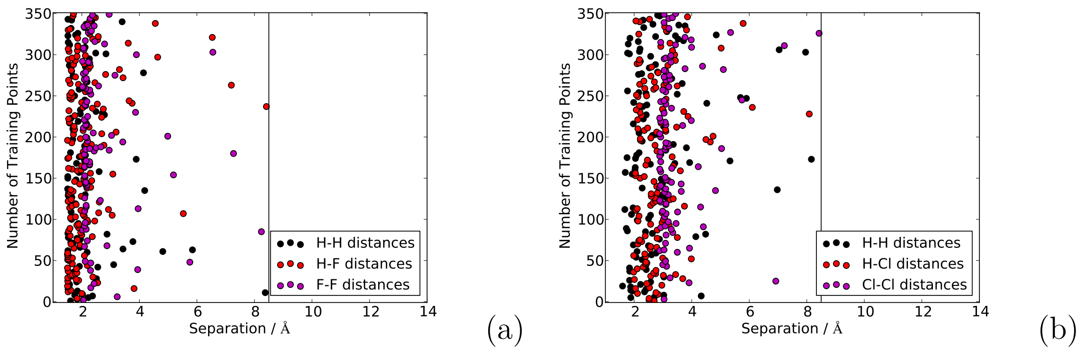


Figure 3.15: Plots showing the training point placement for the HF dimer (a) and HCl dimer (b) for the first 350 training points from the fixed boundary strategy.

erbates the non-stationarity issue. At low separations, the requisite lengthscales for the Cl-Cl and H-H interactions will be vastly different due to the relatively small size of the atoms in the latter. However, at larger separations this will cease to be the case. Consequently, the issue of requiring different lengthscales at different separations is exacerbated for the $(HX)_2$ dimers, specifically $(HCl)_2$.

The effect is not seen under boundary optimisation as this technique ensures that the GP models only a small portion of the repulsive wall when training begins. As this region gradually expands, the hyperparameters selected by the GP gradually change with it. Furthermore, consider the situation where expanding the GP region further would entail simultaneously modelling two portions of the PES for which the optimal hyperparameters significantly differ: any boundary optimisation technique would reject such an expansion as greater predictive accuracy would be achieved by leaving the current long-range region to the long-range function.

The poor performance of the fixed boundary method for these systems renders any comparison of training point placement or predictive accuracy redundant. This is because the strength of the boundary-optimised model is not proven by comparison with a method that performs badly. However, the training point placement of the fixed boundary models may exhibit different behaviour for the $(HX)_2$ systems compared with those in the previous section. Thus, the training points for both of these systems under fixed boundary training are given in figure 3.15.

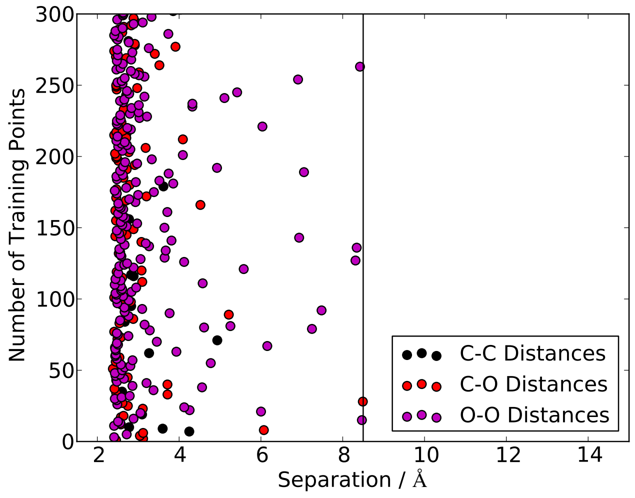


Figure 3.16: Plot showing the training point placement for the CO_2 dimer for the first 300 training points from the fixed boundary strategy.

These plots elucidate no clear difference between the point placement for both systems when compared with that of $(CO_2)_2$ under the same training strategy. This is shown in figure 3.16. However, because the hyperparameters specifying the models fluctuate considerably between stages of training for the $(HX)_2$ systems, the training points are not selected to gradually improve a single set of hyperparameters. This will lead to lower training efficiency. Moreover, that boundary optimisation enhances the stability of hyperparameter selection so completely that any negative effects on training performance are eliminated is itself a promising result.

3.11 Conclusions

It has been shown that boundary optimisation produces GP potentials of the same accuracy using fewer training points than fixing R_{cross} *a priori*. This improvement in efficiency is hierarchical, with a boundary defined by a single, variable crossover distance offering a modest improvement and a boundary defined by multiple such distances facilitating a further gain. Boundary optimisation was also found to enhance stability in the selection of the hyperparameters. In the systems explored in section 3.9, boundary-optimised potentials exhibited higher reproducibility and reduced noise compared with those trained under a fixed R_{cross}.

Meanwhile, for the $(HX)_2$ dimers, boundary optimisation solved a fundamental issue with fixed boundary training, namely large inconsistencies in hyperparameter selection that led to the predictive accuracy of the potential becoming erratic. Under boundary optimisation, the errors of the potentials for these systems fell consistently throughout training.

Generally, the results presented imply that the classifier is more important to the training strategy than the point placement method. In the RMSE range that is suitable for first principles calculations ($\sim$2 x 10^{-5} E_h) the boundary optimisation methods typically reduce the required number of training points by 15-33% relative to a training strategy that is already established as efficient [29]. Because of their reduced training set size, the resulting boundary-optimised PESs are strong candidates for transfer learning, in which the existing *ab initio* calculations are upgraded to a higher level of theory. Moreover, as the size of the GP region increases with the size of the training set only as needed, the resulting potentials are less computationally intensive in applications than fixed boundary methods because they employ the GP over a smaller region of phase space.

The classifier $\mathrm{C}_\mathrm{multi}$, which uses different cross-over distances for difference atomic pairs, performed almost as well as an 'ideal' classifier. Across all systems, the largest difference in performance between the closest model strategy, which uses an 'ideal' classifier, and nearest boundary-optimised strategy was $\sim$ 3 %. In all cases the best-performing boundary-optimised strategy employed $\mathrm{C}_\mathrm{multi}$, implying that a classifier comprising a spherical boundary with a unique radius on each unique atomic pair captures the true boundary effectively. Further refinement of the classifier would not result in a sufficient reduction in training points to justify the extra classifier expense. Furthermore, a complex classifier is less suited to a transfer learning approach: a long-range function that makes accurate predictions in an area of the PES at one level of theory may not do so at another and this issue is exacerbated if the boundary is complicated.

The cross-over distance(s) are learned from the reference data under boundary optimisation using a direct search. This is sufficiently fast to be used at every stage of training whether one or many cross-over distances are employed and in the multi-dimensional case returns cross-over distances in a physically reasonable order. Moreover, both direct search algorithms can be used easily in conjunction with another machine learning technique or on another chemical interaction.

In fact, a direct search can be applied to any problem whereby a boundary is sought between a good method of approximation in one region of input space (the long-range function here) and a machine learning technique in another. This means that the methodology which underpins boundary optimisation is both fast and flexible, in addition to being effective in solving the problem of reducing the computational expense associated with training a GP model of an intermolecular PES.

Physical systems in which the behaviour crosses over from a simple asymptotic function to more complicated behaviour are common in many fields. A prominent example is the transition from ideal to non-ideal gas behaviour. As the boundary optimisation techniques herein exploit this cross-over in behaviour, there are many potential applications of this technique to physical problems beyond intermolecular potentials.

3.12 Future work

The results of training without boundary optimisation on the $(HX)_2$ systems motivates an investigation into the effect of non-stationarity. Such an investigation has been undertaken and is presented in the next chapter.

Furthermore, the application of boundary optimisation to non-additive interactions is likely to show great promise. These potentials are high-dimensional, meaning the possible gains in efficiency are larger than for the systems looked at so far. Additionally, the capability to place different cross-over distances on different interactions may be useful in this context even when the 'mixture' contains a single species. This is because different cross-over distances could be imposed on the shortest, longest and middle distances in an atomic triplet. To do so would require a suitable long-range function, which could be derived by a method similar to that outlined in Appendix D.

Chapter 4

Alternative kernels and transforms for Gaussian process potentials

The results presented in Chapter 3 showed that training without boundary optimisation led to a significant reduction in hyperparameter stability, which in turn reduced training efficiency. While negligible for most systems, for the $(HX)_2$ dimers this effect was pronounced. The instability was rooted in the Gaussian process (GP) having to learn the potential energy surface (PES) out to 8.5 Å from the outset of training, which meant it was trained simultaneously on two regions of the PES where the rate of change in energy with inverse separation differed. That is, the problem occurred due to non-stationarity in the training data.

Consequently, the issue might be alleviated by a more effective transform on the inputs, introduction of a transform on the output (*i.e.* the energy) or by the use of a kernel that is better suited to non-stationary data. These solutions are explored in this chapter. In addition a non-zero mean function could be introduced, with a Lennard-Jones potential a sensible choice. Furthermore, the use of stricter priors could alleviate the issue by forcing the model to prefer one of the two sets of hyperparameters between which the model was seen to vacillate previously. The latter two fixes were not explored here, however.

New input and output transforms, as well as additional kernels, were found to make little difference to training efficiency. However, certain transforms were found to drastically increase the stationarity of the input data. This suggests that re-specifying the training LHCs to the new transforms may enhance training efficiency. Despite this, such work was not undertaken as boundary optimisation already facilitated training that was robust to this issue for these systems. Though fixing this problem within the framework of the GP itself would have been desirable, the option of boundary-optimised training renders it non-urgent. Put simply, the results in this chapter show that certain alterations improved stationarity of the input data but this did not translate to improved training efficiency, as this would require re-specifying the LHCs.

4.1 Methodology

4.1.1 Data sets and systems

The results presented here were all obtained from Latin hypercube (LHC) training, as outlined in section 2.1.4. This training was undertaken on LHCs of sizes 1, 3, 5, 10, 15, 25, 35, 50, 75, 100, 250, 500, 750 and 1000 before application of the high-energy cut-off, $E_{\mathrm{cut}} = 0.005\ E_{\mathrm{h}}$. All calculations were undertaken in Molpro [134]. In previous work, 'moderate' accuracy *ab initio* calculations were used to find and test the best training set design. All calculations in this chapter employ 'moderate' accuracy *ab initio* data, specifically second-order Møller-Plesset perturbation theory (MP2) with an aug-cc-pVTZ basis set and the counterpoise correction. The specifications of the LHCs used for the systems that were investigated are shown in table 4.1. As boundary optimisation is not used, $R_{\mathrm{cross}} = R_{\mathrm{max}} = 8.5$ Å and no predictions were made using a long-range function. That is, all reported RMSE values come from GP predictions only. All GPs were trained using GPy [133] with priors for all hyperparameters being a Gamma distribution with an expected value of one and a variance of two. 20 random restarts were undertaken to optimise the hyperparameters. Results are presented for the $(HCl)_2$ and $(CO)_2$ potentials.

Though it was not examined in the previous chapter, the CO dimer represents an sensible touchstone for the improvements made to training of $(HCl)_2$ poten-

Table 4.1: The co-ordinates for the training and test LHCs for each system. N_{test} is the number of points in the test set after application of E_{cut}. Also shown is the minimum energy in the test set, E_{min}, in E_{h}, though no attempt was made to approximate the global minimum energy for any potential.

System	Coordinate	Range	N_{test}	ϵ
$(CO)_2$	r^{-1}	0.12 to 0.67 Å^{-1}	1519	6.02 x 10^{-4}
	$\cos(\theta_1)$	-1 to 1		
	$\cos(\theta_2)$	-1 to 1		
	ϕ	0 to 180$^{\text{o}}$		
$(HCl)_2$	r^{-1}	0.12 to 0.67 Å^{-1}	1622	2.80 x 10^{-3}
	$\cos(\theta_1)$	-1 to 1		
	$\cos(\theta_2)$	-1 to 1		
	ϕ	0 to 180$^{\text{o}}$		

tial. This is because the $(CO)_2$ PES has the same number of degrees of freedom and symmetries as the $(HCl)_2$ PES and a smaller permanent dipole in its constituent molecules ($\mu_{\text{CO}} \approx$ 0.12 Debye [137] versus $\mu_{\text{HCl}} \approx$ 1.09 Debye [138]). Furthermore, the sizes of its constituent atoms are more similar than those of the $(HCl)_2$ system. Resultantly, any difference between training a $(HCl)_2$ potential and a $(CO)_2$ potential arises from non-stationarity in the training data due to the larger dipole and larger discrepancy in atom sizes in the former. Thus, theoretically, if the issue of non-stationarity were dealt with, the training efficiency of both systems would be near-identical.

That this is not the case under LHC training using a symmetric squared exponential kernel and the $r \to r^{-1}$ transform on the inputs is evidenced in figure 4.1, which shows how $\text{RMSE}_{\text{test}}$ varies with LHC size. Additionally, the figure illustrates that the no-model $\text{RMSE}_{\text{test}}$ values are similar for both systems, with values of 1.18 x 10^{-3} E_{h} and 1.26 x 10^{-3} E_{h} for $(CO)_2$ and $(HCl)_2$ respectively. The no-model RMSEs were calculated by assuming all energy predictions are zero and indicate that the higher errors seen for $(HCl)_2$ are not a consequence of the interaction energies being of a vastly larger magnitude.

Of the $(HF)_2$ and $(HCl)_2$ dimers, the latter was the only one investigated

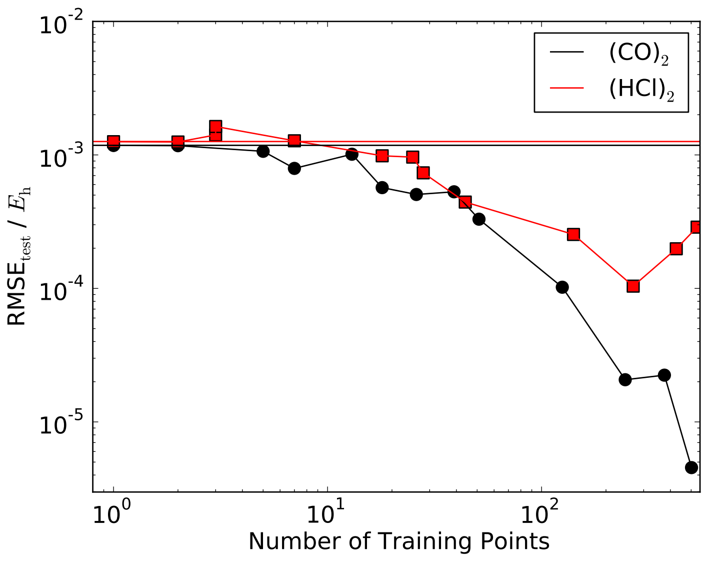


Figure 4.1: Plot of $\mathrm{RMSE}_{\mathrm{test}}$ against LHC size for the $(CO)_2$ and $(HCl)_2$ potentials using k_{sym} as the kernel function and an $r \to r^{-1}$ transform on the inputs. The horizontal lines show the no-model RMSEs, which were 1.18 x 10^{-3} E_{h} and 1.26 x 10^{-3} E_{h} respectively.

here. This is because the results in Chapter 3 illustrate that it was affected more significantly by the effects of non-stationarity than $(HF)_2$. Consequently, any method that reduces the disparity in training efficiency between $(CO)_2$ and $(HCl)_2$ potentials is likely to impact $(HF)_2$ potentials similarly. Furthermore, the larger high-energy cut-off used for the $(HF)_2$ potential makes direct comparison of errors between models of this system and $(CO)_2$ more difficult.

4.1.2 Input transform

As an alternative to the use of inverse interatomic distances, a $r \to (r - ar_{\mathrm{min}})^{-1}$, $a \in [0, 1)$, transform was employed. This transform was chosen as it makes the inverse distances further apart at low r and should make the change in energy in the repulsive wall more gradual as a result. As this effect will be less pronounced effect at the outer edge of the well, the disparity between the requisite hyper-parameters for these regions should be reduced. This is because as r increases, ar_{min} becomes smaller relatively, as does the difference between the $r \to r^{-1}$ and $r \to (r - ar_{\mathrm{min}})^{-1}$ transforms.

In this transform, $r_{\mathrm{min}} \neq R_{\mathrm{min}}$, the minimum distance permitted in the train-

Table 4.2: The values of r_{min} used for the $r \rightarrow (r - ar_{\mathrm{min}})^{-1}$ transform on each interatomic distance.

	$r_{\mathrm{C-C}}$	$r_{\mathrm{C-O}}$	$r_{\mathrm{O-O}}$	$r_{\mathrm{H-H}}$	$r_{\mathrm{H-Cl}}$	$r_{\mathrm{Cl-Cl}}$
r_{min} (Å)	2.62	2.56	2.43	1.61	1.95	2.94

ing LHC. Instead it is the shortest distance present in the test set for a given atomic pair after application of E_{cut}. Thus r_{min} is determined for each atomic pair in the system. The values of r_{min} used for each atomic pair in each system are given in table 4.2. These were chosen by taking the smallest value for each interatomic distance present in the test set.

That r_{min} differs between interatomic interactions means it should alleviate the issue caused by discreapncies in atom sizes, discussed in Section 3.10. This is because ar_{min} will be larger for interactions between larger atoms, making the values of $(r - ar_{\mathrm{min}})^{-1}$ for the H-H and Cl-Cl interactions more similar than those of r^{-1}.

The $r \rightarrow (r - ar_{\mathrm{min}})^{-1}$ transform was trialled by varying a and assessing the concomitant change in training efficiency for the $(\mathrm{CO})_2$ and $(\mathrm{HCl})_2$ potentials. Values of a from 0.5-0.99 were used, as well as $a = 0.1$. Though possible, setting $a = 0$ was not done here as the new transform tends to the $r \rightarrow r^{-1}$ transform as $a \rightarrow 0$. All training was done on LHCs that were built in inverse interatomic distances, not the new transformed coordinates.

An alternative approach is to invoke Morse variables, which have been used with more success in other systems [35, 139]. This entails undertaking a $r \rightarrow \exp(-r/b)$ transform, $b \in [2, 3]$ bohr (1 bohr = 5.29 x 10−11 m). However, Morse variables were not found to outperform inverse distances consistently and the results were for intramolecular PESs where the effects of intermolecular dipole-dipole interactions were not relevant. In addition, it is not obvious how these variables would alleviate the issues that stem from differently-sized atoms. As such, they are not explored here.

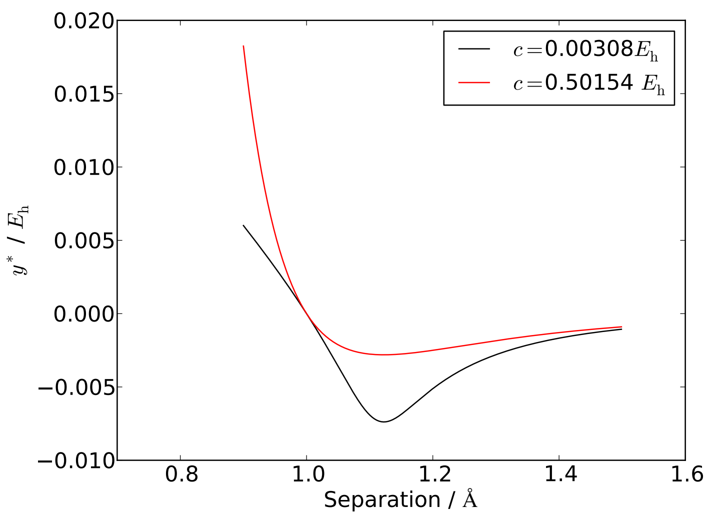


Figure 4.2: A plot showing the Lennard-Jones potential, subject to the energy transform with two different values of c. The original Lennard-Jones potential was specified to have $\sigma = 1$ Å and $\epsilon = 2.8$ x 10^{-3} E_h.

4.1.3 Energy transform

Previously, the impact of transforming the energy before training has not been examined. However, changing the output can, in theory, impart stationarity on the data as effectively as a variation of the inputs. One approach would be to 'squash' the energy values in the repulsive wall to cover a shorter range. This would make the apparent change in energy in this region smaller, increasing the similarity between the rate of change in energy with input in the repulsive wall and the outer edge of the well.

This is the basis of the $E \to c\ln(1 + E/c)$, $c \in [1.1\epsilon, 1]$, transform, where ϵ is the well-depth. The values of ϵ used for each system are given in table 4.1. Here, the well-depth is taken from the lowest energy in the test set. The range of possible values of c arises from the fact the transform is only permitted if $E > -c$.

For positive values of E (*i.e.* those in the repulsive wall), E/c is positive and the transform requires taking the natural log of a value slightly above one. As E grows more positive, the value of $\ln(1 + E/c)$ increases more slowly. The range of the positive, repulsive region of the PES is thereby reduced. Likewise, for negative E, E/c will be negative. Consequently, $1 + E/c < 1$ meaning that the value of the transformed energy will become more negative rapidly with an increase in the magnitude of E. This transform therefore not only truncates the repulsive wall, it elongates the rest of the PES. This should mean that the rate of change in the transformed energy with separation should be closer to uniform than when the energy is not transformed.

The effect of the transform is illustrated in figure 4.2, which shows the Lennard-Jones potential under two different values of c. This illustrates that as $c \to 1.1\epsilon$, the transform changes the energy to a greater extent. This manifests as a deepening of the potential well and a reduction in the range of the repulsive wall. Thus the transform has the aforementioned expected effect and might, therefore, increase the efficiency of training on the $(\mathrm{HCl})_2$ PES.

All $\mathrm{RMSE}_{\mathrm{test}}$ values reported for this transform were obtained by first making the GP predictions in the transformed energy, converting these predictions back to original energies and then comparing with the test set energies. That is, the predictions had to undergo a reverse transform. Denoting the transformed energy as E^*, the reverse transform made prior to evaluating $\mathrm{RMSE}_{\mathrm{test}}$ is

$$E = c\left(\exp\left(\frac{E^*}{c}\right) - 1\right). \tag{4.1}$$

4.1.4 New kernels

Beyond altering the inputs and outputs of the model, non-stationarity in the training data can be addressed with a different kernel. The new kernel employed in this work is the neural network kernel

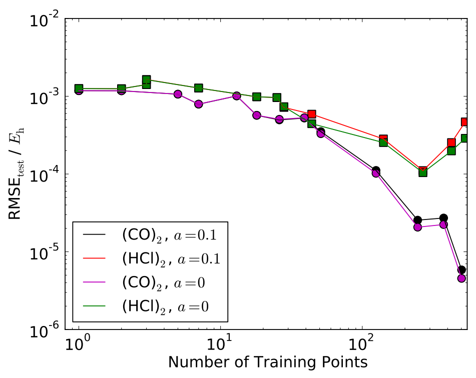
$$k_{\mathrm{NN}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\mathrm{f}}^2 \prod_{d=1}^{D} \frac{2}{\pi} \arcsin\left(\frac{\sigma_{\mathrm{w}}^2 x_i^{(d)} x_j^{(d)} + \sigma_{\mathrm{b}}^2}{\sqrt{\sigma_{\mathrm{w}}^2 (x_i^{(d)})^2 + \sigma_{\mathrm{b}}^2 + 1}\sqrt{\sigma_{\mathrm{w}}^2 (x_j^{(d)})^2 + \sigma_{\mathrm{b}}^2 + 1}}\right) + \sigma_{\mathrm{n}}^2. \tag{4.2}$$

This was chosen due to its non-stationary nature, which will hypothetically account for non-stationarity in the training data.

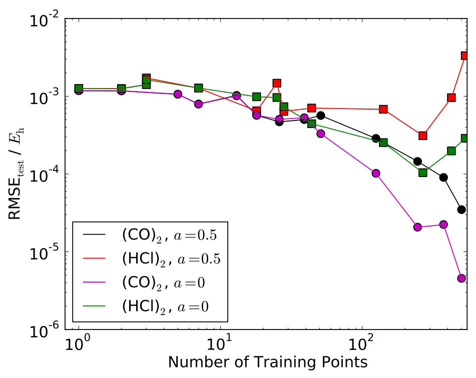
To preserve the symmetric nature of k_{sym}, which allows any interatomic distances in $\mathbf{x}$ to be swapped without affecting the energy, all kernels built from k_{NN} were symmetric. The simplest example is the symmetric version of k_{NN},

$$k_{\mathrm{NN-S}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\mathrm{p} \in \mathrm{P}} k_{\mathrm{NN}}(\mathbf{x}_i, \mathrm{p}\mathbf{x}_j). \tag{4.3}$$

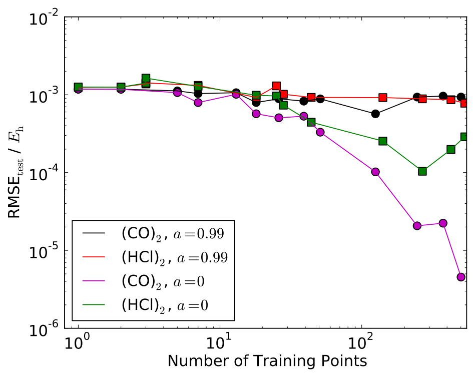
Recall that P is the set of permutations under which the energy is invariant and p is a single permutation within that set.



(a)



(b)



(c)

Figure 4.3: Plots showing the change in RMSE$_{\text{test}}$ with LHC size for the $(CO)_2$ and $(HCl)_2$ potentials that compare the old transform ($a = 0$) with new when $a = 0.1$ (a) $a = 0.5$ (b) and $a = 0.99$ (c).

It has also been found that composite kernels are more efficiently trained for some systems [35]. As such, composite kernels were built by combining k_{NN} and k_{SE} by addition or multiplication. In the later discussion, the most successful of these symmetric composite kernels is discussed. This has the form

$$k_{\text{Comp}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\text{p}\in\text{P}} [k_{\text{NN}}(\mathbf{x}_i, \text{p}\mathbf{x}_j) + k_{\text{SE}}(\mathbf{x}_i, \text{p}\mathbf{x}_j)]. \qquad (4.4)$$

The hyperparameters were optimised by log-likelihood maximisation, not the Bayesian information criterion (see equation (2.21)). As such, the GP model was not penalised for having a large number of hyperparameters. The effect on training efficiency this has would have been investigated had any kernels proved promising in reducing the discrepancy in training $(CO)_2$ and $(HCl)_2$ potentials.

4.2 Results and discussion

Any increase in a for the $r \to (r - ar_{\text{min}})^{-1}$ transform was found to reduce the training efficiency of both potentials. This effect was exacerbated at large a, where the new transform diverged substantially from the old, $r \to r^{-1}$ transform. Such a pattern is evidenced in figure 4.3.

The trend in figure 4.3 suggests that the new input transform inhibits the capacity of the GP to learn either potential as $a \to 1$. While for $a = 0.1$ the results from the old and new transform are similar, with the latter slightly worse, when $a = 0.99$ both potentials barely improve upon the no-model estimates (figure 4.1). Though this corresponds to a reduction in the difference between training GP potentials of the two systems this is only because neither is captured at all well.

However, figure 4.4 shows that the $r \to (r - ar_{\text{min}})^{-1}$ transform does enhance the stationarity of the training data. The figure demonstrates that as a increases, the spread of the data at small separations is larger. Thus, rather than a rapid change in energy at low separation, the change in energy at short-range is far more gradual.

There is no concomitant increase in training efficiency with the increase in the stationarity of the training data, however. Again, figure 4.4 may explain this by illustrating that as a increases the data at smaller separations become more sparse. Thus a LHC designed using the $r \to (r - ar_{\text{min}})^{-1}$ transform would place more points at short range to fill space evenly. As the LHCs were built under the old transform, however, increasing a leads to a training set in which space is no longer filled uniformly. Consequently, training errors increase despite an increase in the stationarity of the training data.

This does, however, suggest that this transform could alleviate the non-stationarity issue when training GP potentials for the $(HX)_2$ systems. To test whether this will lead to an increase in the similarity between training of the $(HX)_2$ and $(CO)_2$ potentials would require re-designing the training LHCs under the new transform. However, provided the training points are dense enough in input space this should not be an issue: the main purpose of designing the LHC in inverse distances is to ensure the density of points in the repulsive wall is large.

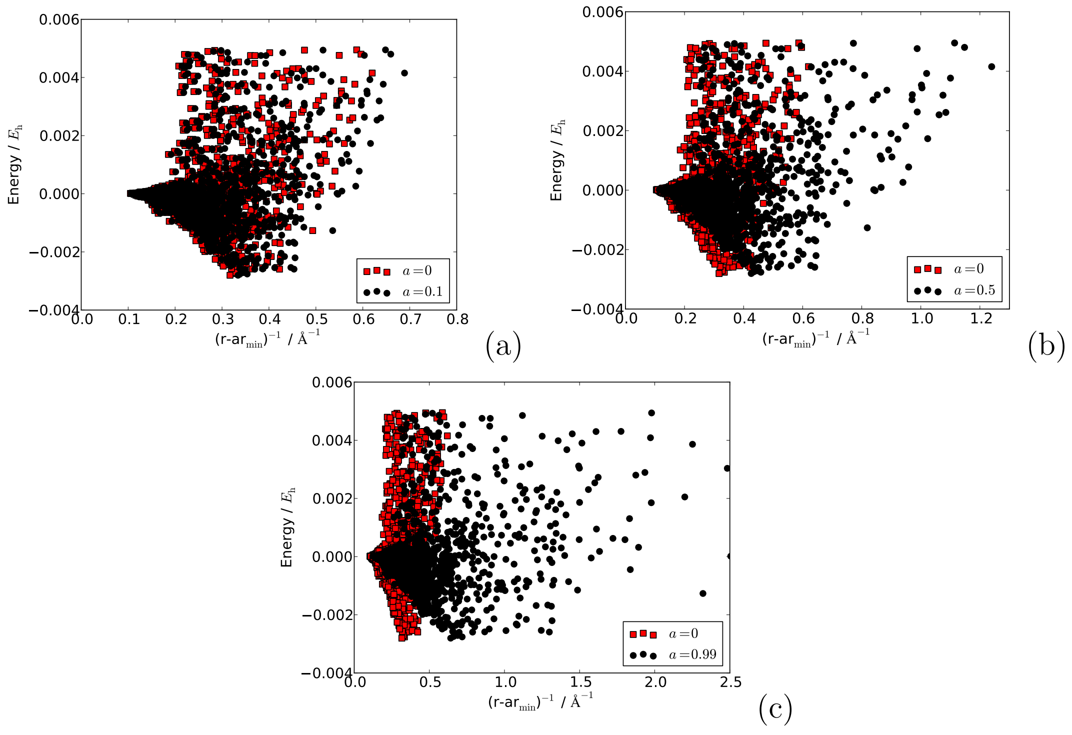


Figure 4.4: Plots showing the distribution of points in the test set for the $(HCl)_2$ potential under the old transform ($a = 0$) and the new when $a = 0.1$ (a) $a = 0.5$ (b) and $a = 0.99$ (c). The third plot is truncated so that points under the original transform are still visible.

Even when transformed again, the number of points in this region is still the same so successful training should still be possible.

The investigation of the $E \to E^*$ transform, meanwhile, began by ascertaining the optimal value of c to use for the $(HCl)_2$ PES. This was done by comparing 300 c values for $c \in [1.1\epsilon, 0.4]$ E_h on the 500-point LHC, for which $\text{RMSE}_{\text{test}}$ was lowest. The results of this comparison are shown in figure 4.5, which reveals that the training error was minimised when $c \approx 0.2302$ E_h.

Though small, it can be seen that this transform leads to a reduction in $\text{RMSE}_{\text{test}}$ for models of the $(HCl)_2$ system. Consequently, GP potentials for the $(CO)_2$ and $(HCl)_2$ systems were trained using $c = 0.02302$ E_h. The results of this training are shown in figure 4.6, which shows that for both systems the reduction in $\text{RMSE}_{\text{test}}$ is minimal for all LHC sizes. This is consistent with the small reduction in $\text{RMSE}_{\text{test}}$ presented in figure 4.5 and is likely a result of the transform being relatively weak. For example, a typical value of the energy would

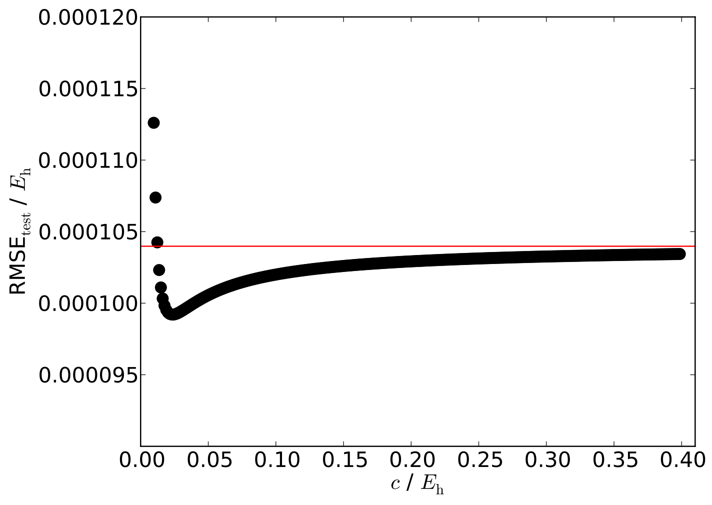


Figure 4.5: A plot showing how RMSE$_{\text{test}}$ varies with the strength of the energy transform for GP potentials trained on the 500-point training LHC. The strength of the energy transform is determined by the parameter c. The red line shows the error when the energy is not transformed. All RMSE$_{\text{test}}$ values were found by comparing GP predictions that had undergone the reverse transform shown in equation (4.1) with the un-transformed test set energies, which were evenly spaced in r^{-1}.

be $E = -1.26 \times 10^{-3}$ E_h, which is the negative of the no-model RMSE. With this energy and $c = 0.02302$ E_h, $E^* = -1.30 \times 10^{-3}$ E_h, corresponding to a difference of only 4×10^{-5} E_h between the transformed and original energies.

However, at $c < 0.2302$ E_h, RMSE$_{\text{test}}$ rises rapidly as the transform becomes more powerful. As a more powerful transform would lead to increased stationarity in the training data, which was alluded to in figure 4.2, one would expect RMSE$_{\text{test}}$ to fall with c. Just as for the input transform though, the increase in stationarity leads to a larger discrepancy between the LHC specified in E and the data set that results from using E^*. Thus training error increases as the input space is filled less evenly as E^* diverges from E.

This is illustrated in figure 4.7, which shows that for the most powerful possible transform (with $c = 1.1\epsilon$ E_h) the placement of points differs more greatly from the original LHC than when $c = 0.02302$ E_h. Thus, once again, the more powerful version of the transform enhances stationarity in the training data while simultaneously leading to a large divergence from the spread of points in the orig-

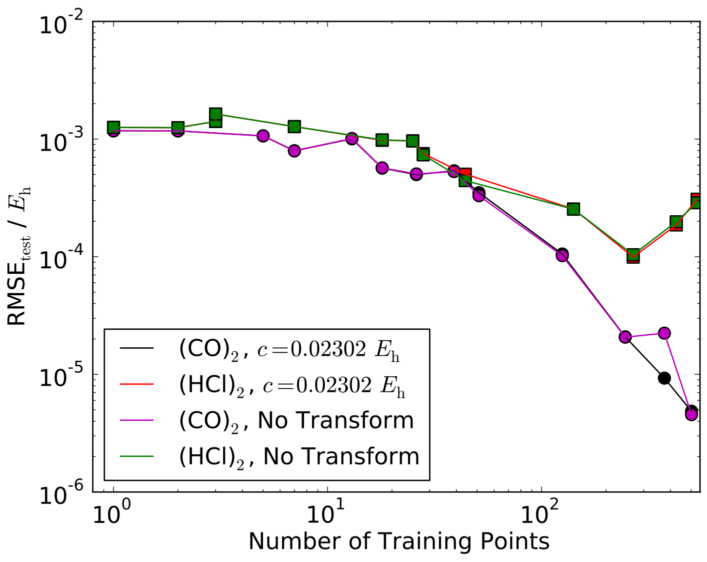
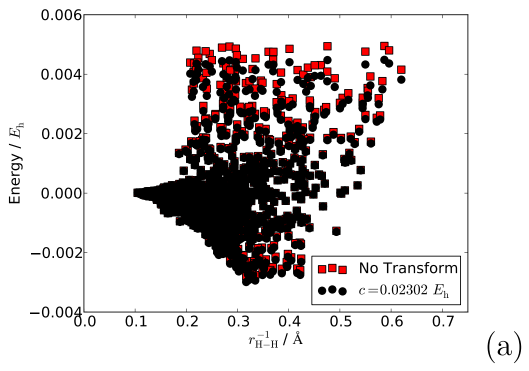


Figure 4.6: A plot showing the change in $\mathrm{RMSE}_{\mathrm{test}}$ with LHC size for the $(\mathrm{CO})_2$ and $(\mathrm{HCl})_2$ potentials that compares error with no energy transform with that achieved with the new transform when $c = 0.02302$.

inal LHC. As the points no longer fill the input space as evenly as they would under a LHC design, the training error increases. Further testing of the transform with training LHCs designed under the transformed energies is therefore required to determine its efficacy, similarly to the $r \rightarrow (r - ar_{\mathrm{min}})^{-1}$ transform.

Figure 4.7 also shows that when $c = 1.1\epsilon$, the difference between the most positive and most negative energies in the test set increases versus the un-transformed data. This will contribute to the reduced accuracy of the GP potentials, as a larger range of energies is modelled with the same number of points that are no longer optimally distributed to fill the input space. Denoting the range of energies as Δ_{T} where T is the transform, $\Delta_{\mathrm{None}} = 7.8 \times 10^{-3}\ E_{\mathrm{h}}$, $\Delta_{\mathrm{c}=1.1\epsilon} = 1.0 \times 10^{-2}\ E_{\mathrm{h}}$ and $\Delta_{\mathrm{c}=0.02302} = 7.5 \times 10^{-3}\ E_{\mathrm{h}}$. This shows that $\Delta_{\mathrm{None}} > \Delta_{\mathrm{c}=0.02302}$, which explains the slight reduction in error at this value of c. The reason that the range of energies expands and then contracts as c decreases is that E^* is only marginally smaller than E initially, with the reduction in the value of positive energies slightly greater than that of the negative energies. However, as $c \rightarrow 1.1\epsilon$ the negative energies continue to decrease rapidly while those that are positive decrease more slowly.

Thus the impact of the above transforms on training efficiency cannot be easily



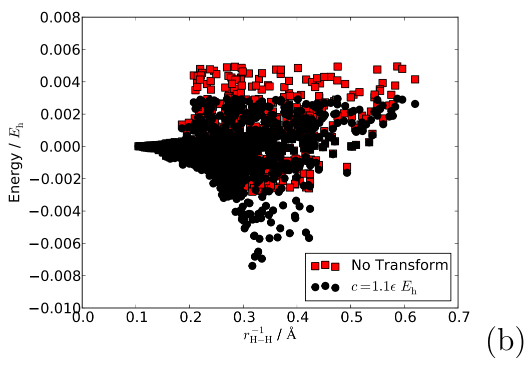


Figure 4.7: Plots showing the distribution of points in the test set for the $(HCl)_2$ potential with no energy transform and the transformed energy when $c = 0.02302$ (a) $c = 1.1\epsilon$ (b).

determined without re-specifying the LHCs used in training. This, however, is not the case when employing a new kernel, which does not change the training data at all. Consequently, any improvement from modifications to the kernel will be reflected in any plot of $\text{RMSE}_{\text{test}}$ against LHC size.

Figure 4.8 therefore contains plots of $\text{RMSE}_{\text{test}}$ against LHC size, demonstrating the reduction in $\text{RMSE}_{\text{test}}$ as the training set grows larger. While this figure evidences that the discrepancy between GP potentials of the $(CO)_2$ and $(HCl)_2$ systems is reduced somewhat by using a symmetric k_{NN}, said improvement is mainly due to errors for the former generally being larger with this kernel than a symmetric k_{SE}. There is, however, a slight improvement over the last two LHCs for the $(HCl)_2$ potential with the new kernel, which may suggest the increased capability with non-stationary training data that it affords is at least partially impactful.

It was hoped that the marginal improvement for the $(HCl)_2$ potential could be preserved and the general decline in accuracy for the $(CO)_2$ potential reversed by using k_{Comp} (equation (4.4)). This kernel combines the capacity to train on non-stationary data of k_{NN} with the proven capability of k_{SE} in building GP potentials of intermolecular interactions.

The training outcomes for this composite kernel are displayed in figure 4.9. The data therein convey that this kernel successfully retains the slight improvement for the $(HCl)_2$ potential while rendering the $(CO)_2$ training near-identical to that seen with a symmetric k_{SE}. However, these data also show that even this

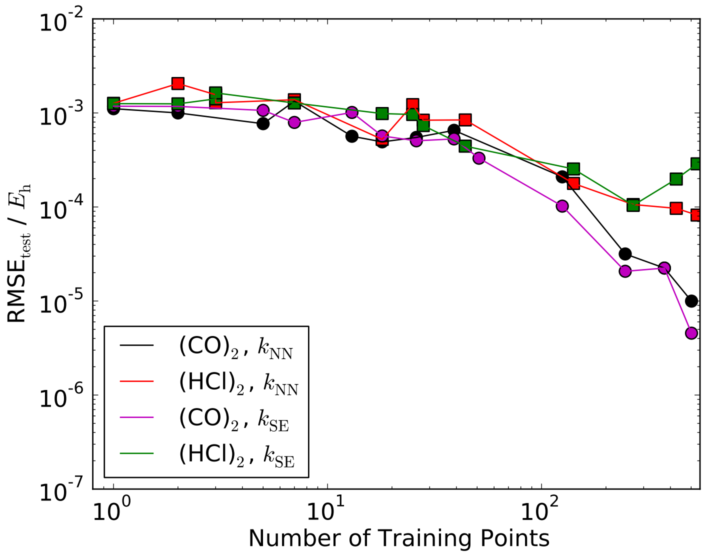


Figure 4.8: A plot of the change in RMSE$_{\text{test}}$ with LHC size for the $(CO)_2$ and $(HCl)_2$ potentials with symmetric versions of k_{NN} and k_{SE}.

kernel offers no significant reduction in the $(HCl)_2$ training errors. Once again, only the RMSE$_{\text{test}}$ values of the final two LHCs are markedly reduced and even then not by a factor sufficient to make training of the two potentials equivalent. Thus, the use of a non-stationary kernel is not sufficient to eliminate the discrepancy between training of $(CO)_2$ and $(HCl)_2$ potentials.

That the introduction of a non-stationary kernel barely results in any improvement to the training errors observed for the $(HCl)_2$ potential also implies that further reducing non-stationarity in the training data will not lead to more efficient training. This is because significant non-stationarity in the data would have meant the improvement accompanying the use of a non-stationary kernel would have been sizeable. Instead, the small improvement observed shows that the $(HCl)_2$ data are only marginally more non-stationary than the $(CO)_2$ data and that the $r \to r^{-1}$ transform is successful in making the data as nearly as stationary as possible. Consequently, further exploration of the aforementioned transforms may not be worthwhile despite the significant changes they impart on the training data. This is because a PES is inherently non-stationary and will remain so under any transform; although these transforms may increase slightly the stationarity of the data in any LHC, the vast majority of the increase is made with the simple $r \to r^{-1}$ transform. Thus any additional gain from a more

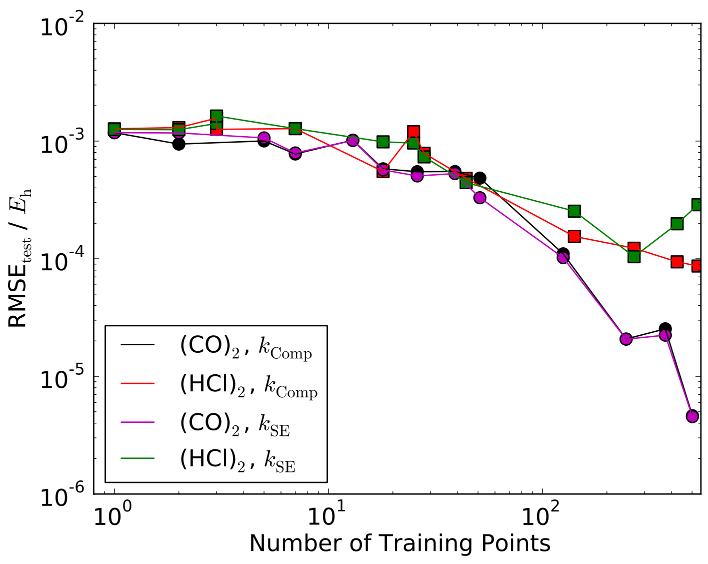


Figure 4.9: A plot of the change in $\mathrm{RMSE}_{\mathrm{test}}$ with LHC size for the $(\mathrm{CO})_2$ and $(\mathrm{HCl})_2$ potentials with k_{Comp} and a symmetric k_{SE}.

convoluted transform will be likely be minor.

4.3 Conclusion

The results in Section 4.2 show that the proposed energy and input transforms alter substantially the distribution of the points in the LHC data sets. This effect is specifically pronounced for the latter transform at large a. However, such changes did not translate in either case to a significant reduction in $\mathrm{RMSE}_{\mathrm{test}}$ for the $(\mathrm{HCl})_2$ system. While the energy transform imparted a marginal reduction in this quantity when $c \approx 0.0203\ E_\mathrm{h}$, the input transform increased the error for all values of a. In both cases, the most powerful transforms resulted in the largest increase in $\mathrm{RMSE}_{\mathrm{test}}$.

The increase in error was the result of the aforementioned alteration to the distribution of points in the training sets under the more powerful transforms. Said change resulted in a less even distribution of points in the new input space than would be the case if the training LHCs had been re-designed under the new transforms. This in turn reduced the efficiency of training, meaning to fully explore the effects of these transforms would require such a re-design to be undertaken.

Despite this, a large enough density of training points should mean that switching between transforms without re-specifying the training LHC should induce no increase in error. Training LHCs are developed in inverse distances to ensure the density of points in the repulsive wall is large, which remains the case even under a new transform. As such, the training LHCs were not re-designed here.

A further reason that the training LHCs were not re-designed here was the observation of the effect of training with a non-stationary kernel. These results implied that, though the data were slightly non-stationary, the residual non-stationarity after the $r \to r^{-1}$ transform had a negligible impact on training. Therefore any further reduction in non-stationarity in the training data would likely result in only minor increases in training efficiency, such as those observed when using a non-stationary kernel.

The results of training with this kernel also showed that training with a composite kernel that stems from adding k_{SE} and k_{NN} preserved the small gain in training efficiency for the $(\mathrm{HCl})_2$ system from the use of k_{NN} while preserving the training efficiency of building the $(\mathrm{CO})_2$ potential. Thus this kernel may be a promising candidate for training GP potentials of more complex PESs. However, the composite kernel has many more hyperparmeters than symmetric versions of k_{SE} or k_{NN}, which renders training more lengthy.

Overall, the results suggest that the non-stationarity that remains in the $(\mathrm{HCl})_2$ training data cannot easily be reduced further, with the original $r \to r^{-1}$ transform seeming to account for the majority of non-stationarity. This is because of the small effect of the non-stationary kernel and the poor results of the new transforms. Instead, the non-stationarity is inherent in the PES, especially the PES of a system such as $(\mathrm{HCl})_2$, which undergoes strong dipole-dipole interactions and features atoms of considerably different sizes.

The results in the last chapter also suggest that this issue is circumvented by the use of boundary optimisation. Therefore, deriving another solution to the problem is not a pressing concern: if one encounters a system comprising differently-sized atoms, any issues with training will likely be alleviated with boundary optimisation anyway.

4.4 Future work

The results in this chapter suggest that alterations to the input and output transforms can drastically alter the distribution of the input data. Consequently, it may be of interest to re-design the LHCs used in training here to follow these new transforms. This would incite more efficient space-filling under the new transforms, which may lead to higher training efficiency of the $(HX)_2$ dimer systems. However, as discussed, this is not necessarily anticipated to improve the results drastically.

Training by sequential design under the new transforms may also be of interest, to ascertain whether selecting a subset of points from a dense reference set would enhance training outcomes. Likewise, training via sequential design using k_{Comp} may be worthwhile for this reason. However, given that the problem is known to be solved by boundary optimisation and has no obvious solution, training with boundary optimisation in future may be the simplest way to avoid it. If a situation were encountered where boundary optimisation offered no significant improvement, then continuing to investigate the effects of new transforms may be worthwhile. However, those already explored here are unlikely to be of use.

Chapter 5

Efficient implementation of Gaussian process potentials

5.1 Chapter overview

A Gaussian process (GP) potential, like any other machine-learned potential (MLP), can be a far more accurate approximation of the true potential energy surface (PES) than that rendered by a traditional force-field. However, though considerably less computationally intensive than using *ab initio* methods directly, MLPs increase the computational cost of simulations relative to force-fields. This problem can be particularly prevalent in determining non-additive contributions, such as the three-body potential. Such potentials require more training points to capture and any simulation box will contain, for example, more triplets than pairs. As such, evaluating efficiently the three-body non-additive potential is the focus of this chapter.

The relatively high cost of GP potentials renders their efficient implementation paramount. Here this is achieved through parallel programming, which entails distributing the individual evaluations that form the total calculation across processes such that they are undertaken simultaneously. Throughout this chapter, a higher efficiency refers to a reduction in the real time required to evaluate a potential.

Efficient implementation represents another route, alongside minimising training set size, to mitigating the increased cost of GP potentials. Some of the techniques discussed here to achieve such an implementation are specific to the architecture of GP potentials. That is, they exploit the framework used by these potentials to make predictions in order to apply them efficiently. Thus, these aspects of the method are not readily generalisable to other MLPs. However, the distribution of the triplet calculations given here is general and effective in ensuring these are disseminated equitably across processes. As a proof of concept, this chapter presents an algorithm to calculate in parallel the three-body contribution to the potential in a simulation box of argon atoms using a GP. The three-body potential is the most computationally intensive calculation that contributes significantly to the total potential. All calculations discussed in this chapter require no approximations, meaning the GP potential is given exactly.

Typically, Gibbs ensemble calculations entail anywhere between 100,000-10,000,000 cycles. Each cycle comprises displacement of all atoms in both boxes, a volume change move and a particle exchange move from each box. Throughout this chapter, a simulation box contains 500 atoms. For such a simulation, if each volume change took one second and each exchange or displacement took 0.1 second, the total time for a cycle would be $\sim$ 102 s. This would leave the total time for the simulation at 10,200,000 s for 100,000 cycles, the equivalent of around 118 days. Thus a ten-fold speed-up could reduce the total time to under two weeks. Consequently, facilitating efficient calculation in simulations to produce results in reasonable time frames is paramount.

The chapter begins with descriptions of the importance of three-body interactions and how to evaluate Ar_3 triplets in Sections 5.2 and 5.3 respectively. This is followed by discussions of how to calculate in parallel all exponentials (Section 5.4 and non-additive energies (Section 5.5). These sections contain descriptions of different calculations required in Monte Carlo simulations and, for ease of discussion, the efficiency of different aspects of the method. A final overview of the speed-up is then given in section 5.8, after descriptions of how to accept and reject moves in Section 5.6, and the current algorithm in Section 5.7. The requisite changes to use GP potentials in simulations of more varied systems are given in section 5.10. This section also contains a brief discussion of the changes needed to apply GP potentials to molecular dynamics simulations, which are important for non-equilibrium properties such as viscosity, and how to evaluate the long-range correction for non-additive interactions. Previously, it was common to set the non-additive energy to zero beyond a certain distance (e.g. 5 Å for the H-O distance in water [140]).

All calculations in this chapter were undertaken on the University of Nottingham high-performance computer. Code was written in Fortran, using the Fortran message passing interface (MPI) for parallelisation and shared memory. For all calculations, the O3 optimiser flag was enabled to minimise computational time prior to parallelisation.

5.2 Significance of the three-body non-additive contribution

Herein there is a particular focus on the three-body contribution (see Chapter 2.3) to the total potential energy. This is because determining the additive contribution requires a simplification of the method used for the three-body calculation and is faster. Thus, any algorithm that can calculate the three-body non-additive energy of a system of particles can be extended straightforwardly to undertake additive energy calculations.

Moreover, the three-body energy is by far the most significant non-additive contribution to the total potential. For water, the two- and three-body contributions account for the vast majority of the potential [141–143]. In fact, these contributions account for on average 95-99 % of the total potential [121–124]. Furthermore, for spherical particles with no permanent dipole such as argon, inclusion of three-body effects is sufficient for accurate Monte Carlo simulations in the liquid phase [131]. It is therefore common for three-body terms to be the highest-order energy contributions in simulations of phase behaviour in noble gases [144, 145]. Thus, a method that is capable of evaluating the additive and three-body energies for many systems, including pure argon, will ignore only contributions generally considered to be negligible. Higher-order many body contributions are neglected here as a result, so the terms “three-body” and “non-additive” are used interchangeably for the duration of this chapter.

In simulations of water the inclusion of these three-body effects improved significantly the results of simulations, drawing the estimates of the number of H-bonds present per molecule [146], the diffusion constant [140], and, among others, the virial coefficients [147] closer to values expected from experiment. Consequently, modern force-fields such as AMOEBA [3–5] employ three-body polarisable terms [6, 7]. This means that to exploit fully the improved accuracy offered by MLPs over traditional force-fields, the former must be able to estimate efficiently the non-additive potential.

If higher-order contributions were desired, the methodology presented here should prove generalisable. The simplest higher-order term is the four-body contribution, which it should be possible to evaluate with relatively minor modifications. For example, the strategy outlined for storing the exponentials would be the same, requiring only a larger array. Likewise, the summation over exponentials would require a similar method, only with an extra exponential in the product and a permutation array that accounts for groups of four atoms rather than triplets. Thus, though it would require a more complex variation of the methodology presented here for the three-body potential, evaluation of the four-body potential should be possible using the same general method.

5.3 Ar_3 triplets

A GP potential for the Ar_3 triplet will require a training set of size N_t, in which each training point comprises a number of dimensions N_d. All GP potentials in this chapter were trained on a 337-point Latin hypercube, with all energies calculated at the CCSD(T) level of theory and an aug-ccPVQZ basis set. That is, $N_\mathrm{t} = 337$ while the number of dimensions per training point, N_d, is three. These potentials also made use of the symmetric squared exponential kernel, which means that the set of permutations under which the energy is invariant was required. This set is introduced later (Section 5.5.1), but as all interatomic distances are equivalent, all can be swapped. Thus the total number of permutations N_perm is six for all triplets.

Throughout this chapter, $N_\mathrm{a} = 500$ and the side length L is 18 Å. A cut-off of $r_{\mathrm{c}} = L/2 = 9$ Å is used, resulting in roughly 12 % of energy calculations being undertaken explicitly. For all cases, half of moves were accepted. Only the additive long-range correction, as shown in 2.4.4, is undertaken here, when evaluating a full simulation box.

5.3.1 Evaluation of the energy of an Ar_3 triplet

Consider a single triplet comprising three Ar atoms, labelled 1, 2 and 3, which is described by the vector of inputs $\mathbf{x} = (x_{12}, x_{13}, x_{23})$. Replacing the positions with these inputs, from the discussion in Chapter 2.3 the total potential energy of this triplet $U_{\mathrm{tri}}(\mathbf{x})$ is given by

$$U_{\mathrm{tri}}(\mathbf{x}) = U_{\mathrm{Add}}(\mathbf{x}) + U_{\mathrm{NA}}(\mathbf{x}), \tag{5.1}$$

where

$$U_{\mathrm{Add}}(\mathbf{x}) = U_{12}(x_{12}) + U_{13}(x_{13}) + U_{23}(x_{23}) \tag{5.2}$$

is the total additive energy of the triplet.

For the rest of this chapter the availability of a trained GP potential is assumed, with the Woodbury vector[1] $\mathbf{\Lambda}$ given by

$$\mathbf{\Lambda} = (\overline{\mathbf{K}} + \bar{\mathbf{I}}\sigma_{\mathrm{n}}^2)^{-1}\mathbf{Y}. \tag{5.3}$$

(See section 2.1.2.) Using a symmetric squared exponential kernel function (equation (2.22)), $U_{\mathrm{NA}}(\mathbf{x})$ is approximated by this GP potential as

$$U_{\mathrm{NA}}(\mathbf{x}) = \sigma_{\mathrm{f}}^2 \sum_{i=1}^{N_{\mathrm{t}}} \Lambda_i \sum_{k=1}^{N_{\mathrm{perm}}} \prod_{j=1}^{N_{\mathrm{d}}} \exp\left(-\frac{(x_j - (x'_{ij})_k)^2}{2l_j^2}\right). \tag{5.4}$$

Here, i sums over training points, k sums over permutations, j runs over dimensions of $\mathbf{x}$, and $(x'_{ij})_k$ is the jth coordinate in the ith training point, after the latter has been subjected to the kth permutation. A permutation is an interchange of two distances under which the energy is invariant. Evaluation of this potential requires $N_{\mathrm{t}}N_{\mathrm{perm}}N_{\mathrm{d}}$ exponentials. As an argon-only simulation will require the same lengthscale for each dimension, henceforth $l_j = l$.

[1]This vector is typically denoted as $\boldsymbol{\alpha}$, but $\mathbf{\Lambda}$ is used here to avoid confusion with α later denoting an atom in a triplet.

Each of the contributions to $U_{\mathrm{Add}}(\mathbf{x})$ are calculated via the same structure as equation (5.4). This calculation will, however, require a different GP potential. Consequently, the values of σ_{f}, Λ_i and l will differ, as will those of N_{perm} and N_{d}. For argon atoms each contribution to $U_{\mathrm{Add}}(\mathbf{x})$ depends only on one distance, so $N_{\mathrm{perm}} = N_{\mathrm{d}} = 1$. Substituting this into equation (5.4) and simplifying yields

$$U_{\mathrm{Add}}(\mathbf{x}) = \sigma_{\mathrm{f}}^2 \sum_{u=1}^{2} \sum_{t=u+1}^{3} \sum_{i=1}^{N_{\mathrm{t}}} \Lambda_i \exp\left(-\frac{(x_{ut} - x_i')^2}{2l^2}\right) \tag{5.5}$$

for the argon additive energy, where the two leftmost sums run over the atoms in the triplet.

5.3.2 Evaluation of an N_{a}-atom simulation box

In any simulation, the total non-additive energy of an entire simulation box U_{NA} is needed. For clarity this discussion begins by considering only a single such box containing N_{a} Ar atoms, which has been placed in a vacuum. Application of periodic boundary conditions and a minimum image convention will be introduced later.

For any value of N_{a}, the simulation box contains $\frac{1}{6}(N_{\mathrm{a}}^3 - 3N_{\mathrm{a}}^2 + 2N_{\mathrm{a}}) \approx N_{\mathrm{a}}^3/6$ separate triplets. Direct evaluation of equation (5.4) over these triplets would require $N_{\mathrm{t}}N_{\mathrm{perm}}N_{\mathrm{d}}N_{\mathrm{a}}^3/6$ exponentials, an expensive potential with an unfavourable scaling with N_{a}.

Similarly, U_{Add} must be evaluated. This involves finding the energies of $(N_{\mathrm{a}}^2 - N_{\mathrm{a}})/2 \approx N_{\mathrm{a}}^2/2$ pairs. Consequently, a direct calculation of U_{Add} would require $N_{\mathrm{t}}N_{\mathrm{perm}}N_{\mathrm{d}}N_{\mathrm{a}}^2/2$ exponentials. Recall that $N_{\mathrm{perm}} = N_{\mathrm{d}} = 1$ for a system of argon atoms, meaning roughly $N_{\mathrm{t}}N_{\mathrm{a}}^2/2$ exponentials must be determined when finding U_{Add}. Despite the $N_{\mathrm{a}}^2/2$ dependence being favourable compared with the $N_{\mathrm{a}}^3/6$ dependence of U_{NA}, the scaling of this calculation with N_{a} is still potentially prohibitive. The more punitive scaling in U_{NA} illustrates that its efficient calculation is paramount.

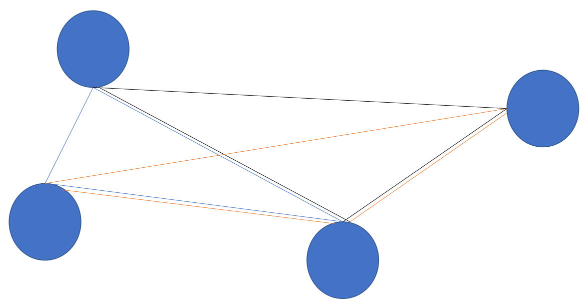


Figure 5.1: A sketch showing how interatomic distances are shared between many triplets.

5.4 Efficient handling of the exponentials

5.4.1 Storage and pre-evaluation

Figure 5.1 highlights that many triplets in the simulation box will share one or more inverse distances. As such, all exponentials can be pre-evaluated and then assembled to calculate U_{NA} for any triplet. That is, although all exponentials will appear in many triplets they need to be calculated once only. This pre-calculation must be undertaken when the box is initialised at the start of a simulation and whenever a volume change move occurs. Similarly, re-evaluation of U_{NA} after a single atom moves or is exchanged only necessitates re-calculation of the exponentials involving that atom. When evaluating U_{Add} for pure argon the second of these observations also applies.

All inverse distances are stored in the matrix $\overline{\mathbf{x}}$, where $x_{\alpha\beta} = \frac{1}{|\mathbf{p}_\alpha - \mathbf{p}_\beta|}$ and $\mathbf{p}_\alpha$ and $\mathbf{p}_\beta$ are atomic positions. Only elements above the main diagonal of this matrix (*i.e.* where $\alpha < \beta$) are of concern as other parts are either self-self distances or known by symmetry. Thus there are $(N_\mathrm{a}^2 - N_\mathrm{a})/2 \approx N_\mathrm{a}^2/2$ unique terms in this matrix and it is assumed hereafter that $\alpha < \beta$. As $\exp\left(-\frac{(x_{\alpha\beta}-x'_{ij})^2}{2l^2}\right)$ must be computed for all atom pairs α, β, the array $\widetilde{\mathbf{E}}$ is introduced. When a triplet comprises three identical atoms as here, $\widetilde{\mathbf{E}}$ is a N_t x N_d x $(N_\mathrm{a}^2 - N_\mathrm{a})/2 \approx N_\mathrm{a}^2/2$ array that contains approximately $N_\mathrm{t}N_\mathrm{d}N_\mathrm{a}^2/2$ exponentials. For any α, β pair, $\widetilde{\mathbf{E}}^{ij}_{\alpha\beta} = \exp\left(-\frac{(x_{\alpha\beta}-x'_{ij})^2}{2l^2}\right)$. However, this array will be larger for a triplet with more permutations (such as $(CO_2)_3$) and smaller for a triplet with fewer permutations

Algorithm 4 Distribution of distances on a single node with shared memory

```
1: Place all $\alpha, \beta$ pairs in a $N_{\text{dists}}$ x 2 matrix, $\overline{\mathbf{Z}}$
2: Find the maximum value, $N_{\max}$, that is less than $N_{\text{dists}}$ and divisible by $N_R - 1$.
3: if $R < R_{\text{top}}$ then
4:     $N_{\exp} = N_{\max}/(N_R - 1)$.
5:     For process $R$, define $S = (R-1)N_{\exp} + 1$ and $F = RN_{\exp}$.
6:     Assign the rows $\overline{\mathbf{Z}}[S : F, :]$ to process $R$.
7: else
8:     $N_{\exp} = N_{\text{dists}}$ - $(N_R - 1)N_{\max}$.
9:     Assign the final $N_{\exp}$ rows of $\overline{\mathbf{Z}}$ to process $R = R_{\text{top}}$.
10: end if
```

(such as Ar-Ne-He).

It was hoped that parallelisation (Appendix F) of the exponential calculation across many processors would increase its speed. However, filling $\widetilde{\mathbf{E}}$ in parallel did not reduce the time taken for the calculation. This is because having each process calculate part of the array and then share it with all others is more time-consuming than having each determine every entry separately. This is due to this approach requiring two steps: the sending of the exponentials from each process to the root process to be stored; and the sharing of the full exponential array from the root process thereafter. Both steps necessitate the distribution of large amounts of data, rendering sharing of the exponentials lengthy.

This cumbersome step can be circumvented by use of shared memory (Appendix F). Shared memory allows processors on the same node to fill part of a shared array that all processors have full access to. Under this method, all atomic pairs are distributed as evenly as possible across the processors on the same node via algorithm 4. This leaves $N_{\exp}$ pairs on each process for which the exponentials must be determined. All parallelisations in this chapter are over a total of N_{p} processors, each with its own rank R, $R \in [1, N_{\text{p}}]$. The process with $R = 1$ is known as the root process. In algorithm 4, meanwhile, N_{dists} is the number of inverse distances for which exponentials must be calculated, $N_{\exp}$ is the number of pairs assigned to a process, R_{top} is the highest-ranked processors on the node, and N_R is the number of processors on the node.

Algorithm 5 Pre-calculation of exponentials with shared memory

```
1: Distribute the distances via algorithm 4.
2: for $R = 1, R_{\text{top}}$ do
3:    for $m = 1, N_{\text{exp}}$ do
4:       Take the $m$th $\alpha, \beta$ pair assigned to process $R$.
5:       for $i = 1, N_{\text{t}}$ do
6:          for $j = 1, N_{\text{d}}$ do
7:             for $k = 1, N_{\text{perm}}$ do
8:                Evaluate $\exp\left(\frac{\left[\text{x}_{\alpha\beta} - \mathbf{x}'_{ij}\right]^2}{2l^2}\right)$.
9:                Save this exponential to $\widetilde{\mathbf{E}}^{ij}_{\alpha\beta}$.
10:            end for
11:         end for
12:      end for
13:   end for
14: end for
```

Once the exponentials for a given pair have been found, the N_{t} x N_{d} segment $\widetilde{\mathbf{E}}_{\alpha\beta}$ that corresponds to that pair is filled. Though this segment is filled by a single process, the data therein is visible to all processors on the same node. Letting $\text{x}_{\alpha\beta}$ be the α, β inverse distance, the method for filling $\widetilde{\mathbf{E}}$ with shared memory on a given node is shown in algorithm 5.

However, processors on different nodes cannot share memory. Therefore, to ensure all processors on a given node have access to a full $\widetilde{\mathbf{E}}$, algorithms 4 and 5 must be followed separately by the processors on each node. For example, consider having four processors labelled $R = 1$ to $R = 4$. If all of these are on a single node, the algorithms are undertaken once by all processors simultaneously. However, if the processors are split into two groups with $R = 1, 2$ on node one and $R = 3, 4$ on node two, each group has to follow the algorithms separately. Consequently, a speed-up of a factor of up to N_{p} is possible if all processors are on a single node, but if the processors are split over many nodes then the speed-up is limited to a factor of R_{top} on the least populous node.

An equivalent array can be built in the same way for the additive energy calculation. Once again, it will contain $(N_{\text{a}}^2 - N_{\text{a}})/2$ sets of exponentials: one for each pair of atoms. However, as this time $N_{\mathrm{d}} = 1$, $\widetilde{\mathbf{E}}$ for the additive interactions can be denoted as $\overline{\mathbf{E}}$, a N_{t} x $(N_{\mathrm{a}}^2 - N_{\mathrm{a}})/2 \approx N_{\mathrm{a}}^2/2$ matrix that contains approximately $N_{\mathrm{t}} N_{\mathrm{a}}^2/2$ exponentials.

5.4.2 Updating the exponentials after an atom move

As discussed in Chapter 2.4, to determine phase co-existence using a Gibbs ensemble requires volume changes, particle displacements and particle exchanges. The previous section describes the methodology required to undertake a volume change, which requires calculation of all exponentials. Currently, no code has been written for exchange moves, so these are discussed in section 5.10. However, such moves are special cases of atom displacements, meaning some insight into exchange moves can be gained from the results for movement of an atom. The following discussion centres on the methods required to re-calculate the exponentials following movement of an atom, which is key to all Monte Carlo simulations.

Following displacement of some atom δ, the exponentials for all pairs involving δ are changed. As a result, the non-additive energies of all triplets containing δ are altered, as are the additive energies of all pairs containing it. The affected exponentials are all

$$\alpha, \delta; \quad \delta > \alpha \tag{5.6}$$

and

$$\delta, \beta; \quad \delta < \beta. \tag{5.7}$$

This leaves a total of $N_{\mathrm{a}} - 1$ pairs for which the inverse interatomic distances have changed. In an exchange move, meanwhile, the box into which the atom is moved will gain the same number of new pairs.

Hence, the first step in evaluating the change in the total non-additive energy ΔU_{NA} is re-calculation of the exponentials corresponding to the $N_a - 1$ affected distances. This is once again done using shared memory, meaning that the $N_{\mathrm{a}} - 1$ affected exponentials must be split between all processors on each node. The number of distances for which a process must re-evaluate the exponentials is denoted N_δ and these distances are assigned to all processors on a single node via algorithm 6.

Algorithm 6 Distribute affected distances with shared memory

```
1: Place all pairs containing $\delta$ in a $N_{\mathrm{a}} - 1$ x 2 matrix, $\overline{\mathbf{W}}$
2: Find the maximum value, $N_{\max}$, that is less than $N_{\mathrm{a}} - 1$ and divisible by $N_R - 1$.
3: if $R < R_{\mathrm{top}}$ then
4:     $N_\delta = N_{\max}/(N_R - 1)$.
5:     For process $R$, define $S = (R-1)N_\delta + 1$ and $F = RN_\delta$.
6:     Assign the rows $\overline{\mathbf{W}}\,[S : F, :]$ to process $R$.
7: else
8:     $N_\delta = N_{\mathrm{a}} - 1 - (N_R - 1)N_{\max}$.
9:     Assign the final $N_\delta$ rows of $\overline{\mathbf{W}}$ to process $R = R_{\mathrm{top}}$.
10: end if
```

As any move that results in an increase in the total energy U_{total} may be rejected, all old exponentials involving δ must be stored. This is done using an array that is not shared between processors, as each process need only save the old exponentials that it is updating. The N_{t} x N_{d} x N_δ array $\widetilde{\mathbf{O}}$ is used to store the old exponentials and $\widetilde{\mathbf{E}}$ remains the only shared array. The necessary steps to update the exponentials under shared memory are shown in algorithm 7.

The exponentials required for finding the change in the additive energy, ΔU_{Add}, are once again found similarly. The old exponentials are stored in a N_{t} x N_δ matrix $\overline{\mathbf{O}}$ on each process but otherwise the steps are identical, apart from the use of the additive exponential matrix $\overline{\mathbf{E}}$.

Algorithm 7 Update changed exponentials with shared memory

```
1: Distribute the changed distances via algorithm 6.
2: for $R = 1, R_{\text{top}}$ do
3:    for $m = 1, N_\delta$ do
4:       Take the $m$th pair assigned to process $R$.
5:       Save the current exponentials for this pair in $\widetilde{\mathbf{E}}^{ij}$ to $\widetilde{\mathbf{O}}[:,:,m]$.
6:       for $i = 1, N_{\text{t}}$ do
7:          for $j = 1, N_{\text{d}}$ do
8:             for $k = 1, N_{\text{perm}}$ do
9:                Evaluate $\exp\left(\frac{\left[\mathrm{x}_{\alpha\beta}-\mathbf{x}'_{ij}\right]^2}{2l^2}\right)$.
10:               Save the new exponential to $\widetilde{\mathbf{E}}^{ij}_{\alpha\beta}$
11:            end for
12:         end for
13:      end for
14:   end for
15: end for
```

5.4.3 Effect of periodic boundary conditions and a minimum image convention

The non-additive energy calculation under the periodic boundary conditions (PBCs) and minimum image convention (MIC) is changed slightly. This is because the interatomic distances are now the distances between the nearest periodic images, which are henceforth referred to as the minimum image (MI) distances, rather than between atoms in the same simulation box. Furthermore, a cut-off distance $r_{\text{c}} \leq L/2$ is introduced alongside these conditions. All triplets in which one or more distances exceed r_{c} have their non-additive energy set to zero, with their contribution instead being approximated with a long-range correction, which is discussed in the future work.

The new definition of the distances changes the exponential calculation. For all atomic pairs assigned to a process in algorithms 4 and 6, the MI distance between the two must be found. This is straightforward, requiring only that β is shifted by the sidelength L through any axis along which the separation between α and β exceeds $L/2$. Consequently, algorithms 5 and 7 still operate as shown with only two differences: the distances in $\overline{\mathbf{x}}$ are now all MI distances and any α, β pair at a separation exceeding r_c has all exponentials set to zero. It is checked if $r_{\alpha\beta} > r_\mathrm{c}$ at stage 4 of algorithms 5 and 7, meaning the total number of exponential evaluations needed is reduced.

Only the exponentials for the MI distances are needed: for a given triplet, if one image of atom 1 is closest to atom 2 and another image closest to atom 3 then either r_{12} or r_{13} will exceed r_c and the energy need not be found explicitly. As the MI r_{12} or r_{13} distances may still appear in other triplets their exponentials are still calculated, but the triplet energy is not. The check of whether a triplet comprises all MI distances is detailed in Section 5.5.6.

5.4.4 Efficiency of calculation using shared memory

The factor by which the speed of a calculation increases when spread over a number of processors N_p is called the speed-up. This is defined as t_1/t_{N_p}, where t_i is the time taken on i processors. As the algorithm for implementing the GP potentials must be capable of evaluating the energy of a full box (for volume change moves) and the change therein after an atom moves (for displacements), speed-ups are presented here for both. Here the term "full box" refers to a full minimum image box, which contains the same number of triplets as a simulation box but all distances are MI distances between the nearest periodic neighbours. For all atom move calculations, the total number of moves, N_moves, is 150. All calculations were undertaken for a 500-atom simulation box of side length $L = 18$ Å. This corresponds to a density of 5689 g dm^{-3}, which means the calculations were in the liquid phase. Shared memory calculations were undertaken using OpenMP.

Figure 5.2 illustrates that, even with shared memory, the exponential calculation does not parallelise particularly well. For both calculations, which include periodic PBCs and an MIC, the speed-up plateaus at roughly a factor of 13-15 after $N_\mathrm{p} > 19$. The red lines in the plots are fitted to the first 10 points, where the speed-up is linear (but not perfectly parallel). They highlight the drastic nature in the reduction in speed-up when the plateau begins.

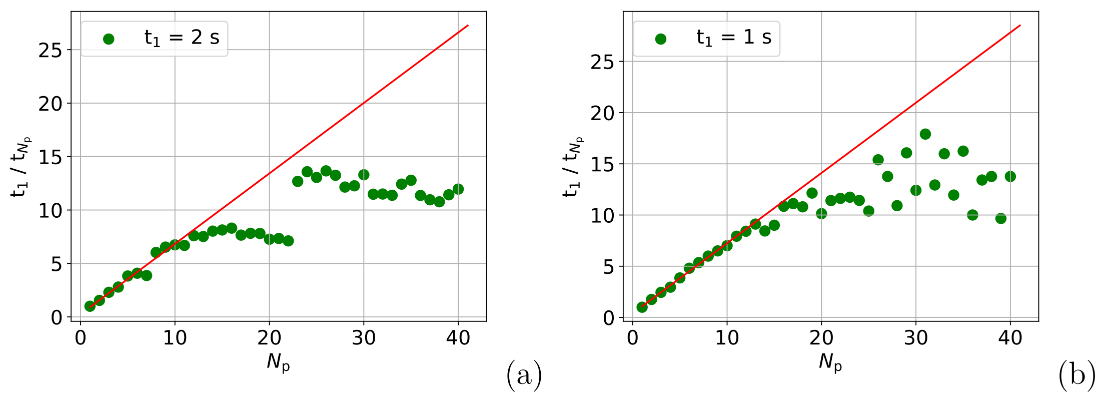


Figure 5.2: A plot showing, over 40 processors on a single node, the speed-up on the root process in the evaluation of the exponentials for the full box calculation (a) and 150 atom move calculations (b).

Two possible causes of the plateau are storage of the old exponentials and accessibility of the shared window to the processors. Currently, exponentials are only stored in the atom move code, which indicates that the former is not the reason. Rather, impeded access to the shared array is likely the root of the issue. This is reinforced by the near-instantaneous transition from linear speed-up to the plateau in both parts of figure 5.2.

However, the poor speed-up in the exponential calculation is not too problematic in practice. This is because, for both types of calculation, the total time taken to fill $\widetilde{\mathbf{E}}$, t_{exp}, is tiny compared to that required to evaluate the triplet energies, t_{triplet}. For the atom move calculation, it is also barely larger than the time needed to sum the non-additive energies, t_{sum}. This is demonstrated for both the full box and atom move calculations in table 5.1.

The times in these tables illustrate that the exponential calculations comprise 11.5 % of t_{total} for the full box calculation and 7.71 % for the atom move calculation. Thus, while improved efficiency in this part of the calculation is desirable, it is not as decisive for performance as improving the efficiency of the triplet energy evaluations. These account for 85.7 % and 86.2 % of the atom move and full box evaluation times respectively.

Table 5.1: Table showing the time taken for different parts of the atom move and full box calculations on a single process, as well as the total time t_{total}. The times for the former are given as an average over 150 moves.

Calculation	t_{total} (s)	t_{exp} (s)	t_{triplet} (s)	t_{sum} (s)
Atom move	0.14	9.80 x 10^{-3}	0.12	9.27 x 10^{-3}
Full box	21.8	2.49	18.8	4.30 x 10^{-2}

5.5 Efficient evaluation of non-additive energies

Having considered the methods for and efficiency of undertaking the exponential calculation, this section describes how the exponentials are combined to evaluate the total non-additive energy $U_{\text{NA}}^{\text{tot}}$. This is a vital part of the algorithm for finding $U_{\text{NA}}^{\text{tot}}$ because it represents the majority of the total time for the simulation, as evidenced in figure 5.1.

5.5.1 Evaluation of the total non-additive energy using the exponential array

Letting γ also denote an atom number, the full range of triplets is described generally as α, β, γ, where $1 \leq \alpha < \beta < \gamma \leq N_{\text{a}}$. Hence the sum over all triplets can be written

$$\frac{U_{\text{NA}}^{\text{tot}}}{\sigma_{\text{f}}^2} = \sum_{\alpha=1}^{N_{\text{a}}-2} \sum_{\beta=\alpha+1}^{N_{\text{a}}-1} \sum_{\gamma=\beta+1}^{N_{\text{a}}} \sum_{i=1}^{N_{\text{t}}} \Lambda_i \sum_{k=1}^{N_{\text{perm}}} \prod_{j=1}^{N_{\text{d}}} \exp\left(-\frac{[(x_{\alpha\beta\gamma})_j - (x'_{ij})_k]^2}{2l^2}\right), \tag{5.8}$$

where $(x_{\alpha\beta\gamma})_j$ is the jth interatomic distance of the triplet α, β, γ and $\mathbf{\Lambda}$ is as shown in equation (5.3). The distances in this triplet comprise the vector $\mathbf{x}_{\alpha\beta\gamma} = (x_{\alpha\beta}, x_{\alpha\gamma}, x_{\beta\gamma})$.

Next, define an operator $\hat{D}_{\alpha,\beta,\gamma,j}$ that computes the relevant interatomic distance for the jth dimension of the triplet α, β, γ. That is

$$\hat{D}_{\alpha,\beta,\gamma,j} = \begin{cases} \alpha\beta & \text{for} \quad j = 1 \\ \alpha\gamma & \text{for} \quad j = 2 \\ \beta\gamma & \text{for} \quad j = 3 \end{cases} \tag{5.9}$$

meaning that $\hat{D}_{\alpha,\beta,\gamma,j}$ returns the jth distance in the α, β, γ triplet when j is specified. This allows the relation of $(x_{\alpha\beta\gamma})_j$ to the array of pre-computed exponentials

$\widetilde{\mathbf{E}}$ through

$$(x_{\alpha\beta\gamma})_j = x_{\hat{D}_{\alpha,\beta,\gamma,j}}. \tag{5.10}$$

While from a mathematical perspective this does little to ease the calculation, it makes coding the energy calculation simpler.

Finally, let $\overline{\mathbf{P}}$ be the matrix of all allowed permutations. For Ar_3 this is

$$\overline{\mathbf{P}} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix}. \tag{5.11}$$

Hence $(x'_{ij})_k = x'_{iP_{kj}}$, which is to say $x'_{iP_{kj}}$ is found by taking the jth element of the kth row in $\overline{\mathbf{P}}$ as the index of the relevant distance in the ith training point.

It follows that

$$\exp\left(-\frac{[(x_{\alpha\beta\gamma})_j - (x'_{ij})_k]^2}{2l^2}\right) = \exp\left(-\frac{[x_{\hat{D}_{\alpha,\beta,\gamma,j}} - x'_{iP_{kj}}]^2}{2l^2}\right) = \mathrm{E}^{iP_{kj}}_{\hat{D}_{\alpha,\beta,\gamma,j}}, \tag{5.12}$$

and $\overline{\mathbf{E}}_{\hat{D}_{\alpha,\beta,\gamma,j}}$ is a N_{d} x N_{t} matrix from within $\widetilde{\mathbf{E}}$ containing all exponentials pertinent to the jth distance in the α,β,γ triplet. Once more, this relation does not simplify things mathematically but does facilitate simpler implementation. Thus the sum over all triplets becomes

$$\begin{aligned} \frac{U^{\mathrm{tot}}_{\mathrm{NA}}}{\sigma^2_{\mathrm{f}}} &= \sum_{\alpha=1}^{N_{\mathrm{a}}-2} \sum_{\beta=\alpha+1}^{N_{\mathrm{a}}-1} \sum_{\gamma=\beta+1}^{N_{\mathrm{a}}} \sum_{i=1}^{N_{\mathrm{t}}} \Lambda_i \sum_{k=1}^{N_{\mathrm{perm}}} \prod_{j=1}^{N_{\mathrm{d}}} \overline{\mathbf{E}}_{\hat{D}_{\alpha,\beta,\gamma,j}}, \\ &= \sum_{\alpha=1}^{N_{\mathrm{a}}-2} \sum_{\beta=\alpha+1}^{N_{\mathrm{a}}-1} \sum_{\gamma=\beta+1}^{N_{\mathrm{a}}} \sum_{i=1}^{N_{\mathrm{t}}} \Lambda_i \sum_{k=1}^{N_{\mathrm{perm}}} \mathrm{E}^{iP_{k1}}_{\alpha\beta} \mathrm{E}^{iP_{k2}}_{\alpha\gamma} \mathrm{E}^{iP_{k3}}_{\beta\gamma}. \end{aligned} \tag{5.13}$$

For example, for a given training point when $k = 4$, the $\mathrm{E}^{iP_{k1}}_{\alpha\beta} \mathrm{E}^{iP_{k2}}_{\alpha\gamma} \mathrm{E}^{iP_{k3}}_{\beta\gamma}$ term in equation (5.13) involves taking the product of the exponential between the $\alpha - \beta$ inverse distance and the second inverse distance in the training point, the exponential between the $\alpha - \gamma$ inverse distance and the third such distance in the training point, and the exponential between the $\beta - \gamma$ inverse distance and the first inverse distance in the training point. The rightmost sum in this equation therefore entails finding these products for all rows of the permutation matrix and adding them.

5.5.2 Evaluation of the total additive energy from the additive exponential matrix

Equivalently to equation (5.8), the total additive energy is given by

$$\frac{U_{\mathrm{Add}}}{\sigma_{\mathrm{f}}} = \sum_{\alpha=1}^{N_{\mathrm{a}}-1} \sum_{\beta=\alpha+1}^{N_{\mathrm{a}}} \sum_{i=1}^{N_{\mathrm{t}}} \Lambda_i \exp\left(-\frac{(x_{\alpha\beta} - x'_i)^2}{2l^2}\right). \tag{5.14}$$

Equation (5.14) is obtained by generalising equation (5.5) to N_a atoms.

As additive energies in this example rely on a single distance only, there is no need for an additive equivalent to $\hat{D}_{\alpha,\beta,\gamma,j}$. Similarly, $N_{\mathrm{perm}} = 1$ here, so there is no need for a permutation matrix in this case. It follows that

$$\exp\left(-\frac{(x_{\alpha\beta} - x'_i)^2}{2l^2}\right) = \mathrm{E}^i_{\alpha\beta}, \tag{5.15}$$

and $\mathbf{E}_{\alpha\beta}$ is a vector of exponentials between $x_{\alpha\beta}$ and all training points. Thus one can re-write equation (5.13) as

$$\frac{U_{\mathrm{Add}}}{\sigma_{\mathrm{f}}} = \sum_{\alpha=1}^{N_{\mathrm{a}}-1} \sum_{\beta=\alpha+1}^{N_{\mathrm{a}}} \sum_{i=1}^{N_{\mathrm{t}}} \Lambda_i \mathrm{E}^i_{\alpha\beta}. \tag{5.16}$$

This highlights once more that evaluating the additive energy requires a simplification of the method used to determine the non-additive energy.

5.5.3 Parallel evaluation of triplet energies

U_{NA} is a sum over the non-additive energies of all triplets, such as in the third term of equation (2.52). Consequently, parallel programming is used to spread the calculation of U_{NA} over N_{p} processors. This reduces the number of triplet evaluations undertaken on each process by up to a factor of N_{p}. As each process need only send back the total energy of all triplets it was assigned (*i.e.* a single number), the issue of cumbersome information sharing that beset the exponential calculations is not an issue. All of this is also true of the additive energy calculation.

Before determining which triplets are calculated by which processors, all atom pairs in the simulation box are divided between the processors. The pairs assigned to each process are consistent throughout the calculation. Conveniently, assigning the pairs in this way is also conducive to the additive energy calculation: each process must simply determine the additive energy across all pairs it is assigned.

It must be ensured that no process possesses all pairs containing a given atom, otherwise moving that atom will leave one process to undertake the entire calculation. To avoid this scenario, the pairs are divided via algorithm 8. In practice, N_{a} will usually be far larger than N_{p}. Consequently, it is assumed that all processors will need access to all exponentials to evaluate their part of the total non-additive energy.

This approach for distributing the pairs across the processors is general to all methods of predicting the energy. Thus it represents an effective route to parallelising the energy calculation whether or not GP potentials are used for prediction. Furthermore, it is well-suited to the evaluation of the non-additive energy as each process need only calculate the energies of all pairs it is assigned, sum these energies and return this value to the root process.

5.5.4 Calculation of the total energy

Following assignment of the atom pairs, each process must determine the non-additive energies of all triplets α, β, γ, where $\alpha < \beta < \gamma$ by definition, for which it owns the α, β pairs. The individual non-additive triplet energies are stored in a vector $\mathbf{U}$, which has length equal to the number of triplets, N_{tri}, assigned to

Algorithm 8 Distribution of atomic pairs across processors

```
1: Assign $N_\mathrm{p}$ pairs to the processors in ascending order; e.g. for the first $N_\mathrm{p}$
   pairs use the loop:
2: for $R = 1, N_\mathrm{p}$ do
3:     Assign pair $1, R+1$ to process $R$.
4: end for
5: Assign the next $N_\mathrm{p}$ pairs to the processors in descending order; e.g. for the
   second $N_\mathrm{p}$ pairs use the loop:
6: for $i = 1, N_\mathrm{p}$ do
7:     Assign pair $1, N_\mathrm{p}+1+i$ to process $R = N_\mathrm{p}+1-i$.
8: end for
9: Once the first $N_\mathrm{a}-1$ pairs are assigned (i.e. all pairs involving atom 1),
   continue assigning pairs in the current order (e.g. continue in ascending order
   if already assigning in ascending order) until all processors are given a new
   pair.
10: Begin assigning pairs in the opposite order to step 6.
11: Repeat steps 1, 5, 6 and 7 until all pairs are assigned.
```

the process. The steps undertaken for the parallel calculation of U_NA are given in algorithm 9, ignoring for now the PBCs and MIC.

In other words, each process undertakes a portion of the sum in equation (5.13), with the three outer sums (over α, β and γ) distributed among them. Due to the manner in which the pairs are distributed, some processors may possess more triplets than others, though not to a significant extent. For example, when sharing the triplets for a 500-atom simulation box over 15 processors, the number of triplets on the most sparse process is 98.18 % of the number on the most populous process. (1369208 is the minimum number and 139463 the maximum number of calculations performed.) Furthermore, because the pairs assigned to a process are the same throughout the simulation, so are the triplets it will evaluate.

Finding U_Add requires a simplified version of algorithm 9. The inner loop (over γ) is no longer necessary and neither is the 'if' statement. Instead, each process simply determines the additive energy of all pairs it is assigned using equation (5.14). These energies are subsequently added and returned to the root, which adds together all of these partial sums.

Algorithm 9 Parallel calculation of $U_{\mathrm{NA}}^{\mathrm{tot}}$ by process R

```
1: Assign a number of $\alpha, \beta$ atom pairs $N_{\mathrm{pairs}}$ to a process by following the steps
   from 5.5.3.
2: for $i = 1, N_{\mathrm{pairs}}$ do
3:     Take the $i$th $\alpha, \beta$ pair assigned to the process.
4:     for $\gamma = \beta + 1,\ N_a$ do
5:         Calculate non-additive energy for the $\alpha, \beta, \gamma$ triplet via the two innermost
           sums in the second line of equation (5.13).
6:         Store the energy from the last step in $\mathbf{U}$.
7:     end for
8: end for
9: Sum over the elements of $\mathbf{U}$ to find the contribution to $U_{\mathrm{NA}}^{\mathrm{tot}}$ from the triplets
   on this process.
10: Send this contribution to the root process.
11: if $R = 1$ then
12:     Add all the non-additive contributions from all processors to get $U_{\mathrm{NA}}^{\mathrm{tot}}$.
13: end if
```

5.5.5 Calculation of the energy following an atom move

Once the exponentials have been updated (see subsection 5.4.2) they are used to determine the change in energy on each process after an atom δ has moved. For any process, the non-additive component of that energy is denoted $\Delta U_{\mathrm{NA}}^{(R)}$, where R is the process rank. The value of $\Delta U_{\mathrm{NA}}^{(R)}$ after move m is

$$\Delta U_{\mathrm{NA}}^{(R)} = U_{\mathrm{NA}}^{(R)} \left(\overline{\mathbf{x}}_{\alpha\beta}^{(m)}\right) - U_{\mathrm{NA}}^{(R)} \left(\overline{\mathbf{x}}_{\alpha\beta}^{(m-1)}\right), \tag{5.17}$$

where $\overline{\mathbf{x}}_{\alpha\beta}^{(m)}$ is the set of inverse interatomic distances after move m. The effects of the PBCs and MIC on this calculation will once again be introduced later, so the distances in $\overline{\mathbf{x}}_{\alpha\beta}^{(m)}$ are not necessarily minimum image distances. The total change in the non-additive energy is given by

$$\Delta U_{\mathrm{NA}} = \sum_{R=1}^{N_{\mathrm{p}}} \Delta U_{\mathrm{NA}}^{(R)}. \tag{5.18}$$

Equation (5.17), however, implies that $U_{\mathrm{NA}}\left(\overline{\mathbf{x}}_{\alpha\beta}^{(m)}\right)$ must be re-calculated in full following a move. This is not the case. Instead, the difference between the new and old non-additive energies for each triplet containing δ is calculated on each process. These differences are summed subsequently to give $\Delta U_{\mathrm{NA}}^{(R)}$.

To that end, the new non-additive energies of each affected triplet on a given process are stored in the vector $\mathbf{C}^{\mathrm{new}}$. Equivalently, the vector $\mathbf{C}^{\mathrm{old}}$ contains the old energies of the same triplets. Both vectors have length equal to the number of affected triplets on the process, N_{changed}. This gives

$$\Delta U_{\mathrm{NA}}^{(R)} = \sum_{i=1}^{N_{\mathrm{changed}}} \left(\mathbf{C}_i^{\mathrm{new}} - \mathbf{C}_i^{\mathrm{old}}\right). \tag{5.19}$$

The steps taken on each process to identify affected triplets and re-calculate their energies are shown in algorithm 10.

Algorithm 10 Parallel calculation of ΔU_{NA} by process R after movement of atom δ

```
 1: for $i = 1, N_{\text{pairs}}$ do
 2:   if $\alpha < \beta < \delta$ then
 3:     Store the old non-additive energy of $\alpha, \beta, \delta$ in $\mathbf{C}^{\text{old}}$.
 4:     Calculate the new non-additive energy for $\alpha, \beta, \delta$ and store in $\mathbf{C}^{\text{new}}$.
 5:   else if $(\alpha = \delta) < \beta$ then
 6:     for $\gamma = \beta + 1,\ N_a$ do
 7:       Store the old non-additive energy of $\delta, \beta, \gamma$ in $\mathbf{C}^{\text{old}}$.
 8:       Calculate the new non-additive energy for $\delta, \beta, \gamma$ and store in $\mathbf{C}^{\text{new}}$.
 9:     end for
10:   else if $\alpha < (\beta = \delta)$ then
11:     for $\gamma = \beta + 1,\ N_a$ do
12:       Store the old non-additive energy of $\alpha, \delta, \gamma$ in $\mathbf{C}^{\text{old}}$.
13:       Calculate the new non-additive energy for $\alpha, \delta, \gamma$ and store in $\mathbf{C}^{\text{new}}$.
14:     end for
15:   else if $\alpha < \gamma < \beta$ or $\gamma < \alpha < \beta$ then
16:     Do nothing.
17:   end if
18: end for
19: Determine $\Delta U_{\text{NA}}^{(R)}$ using equation (5.19) and send to the root.
20: if $R = 1$ then
21:   Evaluate $\Delta U_{\text{NA}}$ using equation (5.18)
22: end if
```

The total change in energy on each process, $\Delta U^{(R)}$, is required, where

$$\Delta U^{(R)} = \Delta U_{\text{NA}}^{(R)} + \Delta U_{\text{Add}}^{(R)}. \tag{5.20}$$

Thus the additive energy change on each process $\Delta U_{\text{Add}}^{(R)}$ must also be found. This calculation is detailed in algorithm 11.

Algorithm 11 Parallel calculation of ΔU_{Add} by process R after movement of atom δ

```
1: for $i = 1, N_{\text{pairs}}$ do
2:     if $(\alpha = \delta) < \beta$ then
3:         Store the old additive energy of $\delta, \beta$.
4:         Calculate the new additive energy for $\delta, \beta$ and store.
5:     else if $\alpha < (\beta = \delta)$ then
6:         Store the old additive energy of $\alpha, \delta$.
7:         Calculate the new additive energy for $\alpha, \delta$ and store.
8:     end if
9: end for
10: Determine $\Delta U^{(R)}_{\text{Add}}$ from the differences between the new and old energies and
    send to the root.
11: if $R = 1$ then
12:     Evaluate $\Delta U_{\text{Add}}$ using equation (5.18) with the additive rather than non-
        additive energies.
13: end if
```

5.5.6 Impact of periodic boundary conditions and a minimum image convention

Evaluating the non-additive energy under the PBCs and MIC is somewhat less straightforward than the exponentials. Not only must it be checked that all distances in the triplet are less than r_{c}, it must be ensured that the minimum image (MI) distances have all atoms in the same positions. For example, if the MI distance between atoms α and β has β placed at (x,y,z) while the MI distance between β and γ has β at (x',y',z') the triplet will have an atom occupying two different positions at once. In this example, having β at (x,y,z) means that $r_{\beta\gamma} > r_{\text{c}}$ and the explicit triplet energy must be zero.

The two aforementioned checks are detailed in algorithm 12, where r_{min} and r_{max} are the the minimum and maximum distances in the α, β, γ triplet. The checks in this algorithm are carried out prior to any energy calculation. This means the PBCs and MIC reduce the total number of energy evaluations, thereby

Algorithm 12 Check triplet distances under PBCs and MIC.

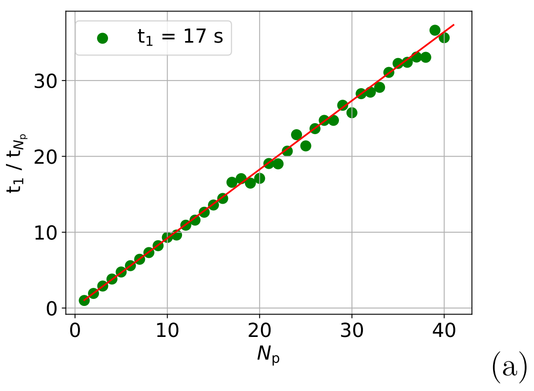
```
1:  if $r_{\max} > r_c$ then
2:      $U_{\mathrm{NA}} = 0$.
3:  else
4:      Keep the atom with the lowest index in $r_{\min}$ in its current position (e.g. if
        $r_{\min} = r_{\alpha,\beta}$ then leave $\alpha$ unmoved).
5:      Move the other two atoms to their minimum image positions relative to
        the unmoved atom.
6:      Calculate the distance $r'$ between the two moved atoms in their current
        positions.
7:      if $r' > r_c$ then
8:          $U_{\mathrm{NA}} = 0$.
9:      else
10:         Calculate the non-additive energy via the two innermost sums on the
            second line of equation (5.13).
11:     end if
12: end if
```

reducing the total time taken for any simulation.

5.5.7 Efficiency of the parallel non-additive energy calculation

It was established in subsection 5.4.3 that the majority of the calculation time stems from the summation of the exponentials to yield the non-additive energies of the triplets. Consequently, it must be verified that the time taken for this portion of the implementation, t_{triplet}, parallelises well. Once again, all references to a "full box" calculation concern a full minimum image box. Just as for the exponential calculations, all calculations were undertaken for a 500-atom simulation box of side length $L = 18$ Å. Therefore the density is again 5689 g dm^{-3} and all calculations were in the liquid phase.

Figure 5.3 illustrates that for evaluation of both the total non-additive energy and the change therein following an atom displacement, t_{triplet} parallelises



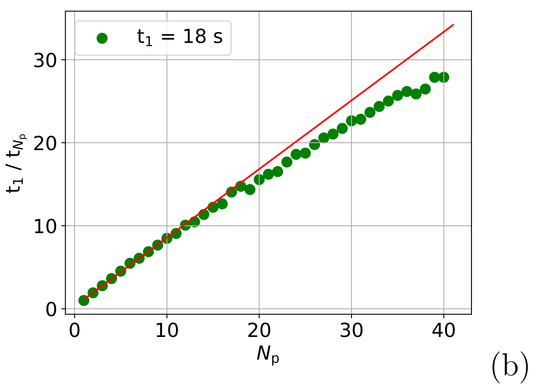


Figure 5.3: Plots showing the speed-up on the root process for the evaluation of the change in the non-additive energy after an atom is moved (a) and the total non-additive energy (b). Both plots are over 40 processors on one node, while the data in (a) are averaged over 150 moves.

near-perfectly. The red lines in both plots are fitted to the first 10 data points where the speed-up is linear and only marginally worse than perfectly parallel. In figure 5.3(a) there is no deviation from this line up to $N_\mathrm{p} = 40$ for the atom move calculation, indicating totally linear (if not quite perfect) parallel speed-up. Meanwhile, figure 5.3(b) illustrates initial linearity with a slight deviation at higher N_p.

The speed-up in t_triplet in both calculations is therefore far better than that seen in t_exp, which is promising given the significant contribution of the former to total calculation time. Moreover, the calculation of the non-additive energy from the exponentials does not require shared memory. Thus it can be spread across processors on multiple nodes with the expectation that the speed-up will continue to increase.

This is evidenced in figure 5.4, which shows equivalent plots over 120 processors divided across four nodes. The calculations are distributed such that the processors are shared evenly over the nodes. For example, for $N_\mathrm{p} = 120$, there are $120/4 = 30$ processors per node.

The data in this figure illustrate a greater speed-up in the evaluation of ΔU_NA after a displacement than in U_NA, which is in agreement with the data in figure 5.3. The rate of speed-up for both calculations with the larger number of nodes is slightly reduced, though the overall speed-up achieved is significant in both

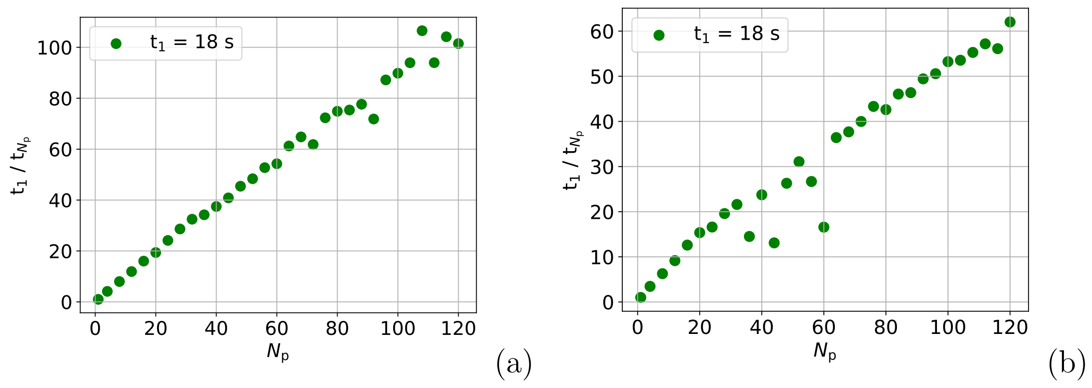


Figure 5.4: Plots showing the speed-up on the root process for the evaluation of the change in the non-additive energy after an atom is moved (a) and total non-additive energy (b). Both plots are over 120 processors on four nodes. The data in (a) are averaged over 150 moves, though t_1 (top left) is the total time over all moves.

cases.

Figure 5.4 shows that a 100-fold speed-up is achieved for the calculation of ΔU_{NA} and a 60-fold speed-up for U_{NA}. This is indicative of a significant reduction in computational time, corresponding to roughly 0.18 s for 150 evaluations of ΔU_{NA} and 0.3 s for a single evaluation of U_{NA}. Thus, the algorithms outlined earlier offer a promising route to the efficient evaluation of the non-additive energy in a simulation with a GP potential.

5.6 Accepting and rejecting moves

In any Monte Carlo (MC) simulation, a move which results in a negative ΔU will always be accepted, while one which results in a positive ΔU can be either accepted or rejected. When a move is accepted, the methods outlined above require saving the new triplet non-additive energies, exponentials, pairwise additive energies, U, the position of the moved atom, and the inverse distances. Meanwhile, rejecting a move necessitates replacing the newly calculated exponentials with the old exponentials saved in $\widetilde{\mathbf{O}}$ and $\overline{\mathbf{O}}$, though U, the position of the moved atom, the inverse distances and $\mathbf{C}^{\mathrm{old}}$ need not be updated. The steps followed on each process to update the data after an accepted or rejected move are given in algorithm 13, which pertains to the non-additive energy only.

Algorithm 13 Move data updates

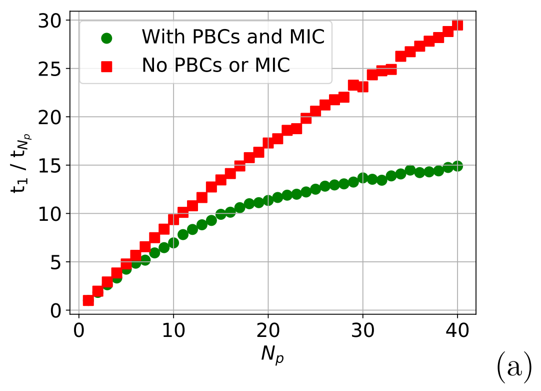
```
1: if The move is accepted then
2:     Update $U_{\text{NA}}$.
3:     Update the distances in $\overline{\mathbf{x}}_{\alpha\beta}$.
4:     Update the triplet energies with the values in $\mathbf{C}^{\text{new}}$.
5:     Update the position of atom $\delta$.
6:     Save the updated exponentials.
7: else
8:     Reset all affected exponentials in $\widetilde{\mathbf{E}}$ with the old exponentials saved in $\widetilde{\mathbf{O}}$.
9: end if
```

5.7 Full algorithm for the non-additive energy calculation

The methods presented in the preceding discussion are summarised below in algorithm 14, which describes how the non-additive energies are calculated for an entire simulation over N_{moves} moves. This algorithm includes all considerations of the PBCs and MIC.

Algorithm 14 Full algorithm for calculating U_{NA}

```
1: Find $\overline{\mathbf{x}}_{\alpha\beta}$ from the minimum image distances.
2: Pre-calculate $\widetilde{\mathbf{E}}$ using algorithm 5 for all pairs for which the interatomic distance $r$ is less than or equal to $r_{\text{c}}$ and set all others to zero.
3: Evaluate $U_{\text{NA}}$ via algorithm 9 and check whether to calculate the energy explicitly using algorithm 12 for each triplet.
4: for $i = 1, N_{\text{moves}}$ do
5:     Find the values of the changed minimum image distances.
6:     Update the affected exponentials using algorithm 7 for all pairs where $r \leq r_{\text{c}}$.
7:     Determine $\Delta U_{\text{NA}}$ via algorithm 10 and check whether to calculate the energy explicitly using algorithm 12 for each triplet.
8:     Update the data according to algorithm 13.
9: end for
```



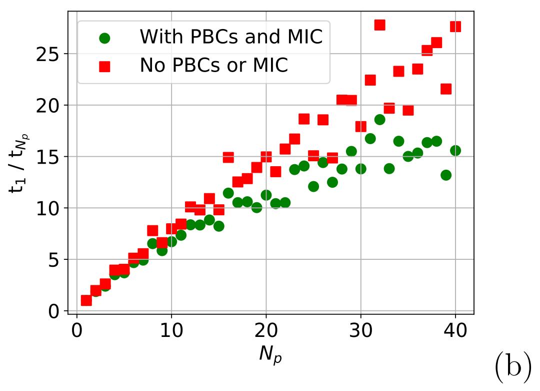


Figure 5.5: A plot showing, over 40 processors on one node, a comparison of the speed-up on the root process in the calculation of the energy without PBCs and an MIC (red) and with PBCs and an MIC (green) for the full box calculation (a) and the atom move calculation (b).

5.8 Results and discussion

Thus far, the efficiencies of the exponential and energy calculations have been discussed in subsections 5.4.4 and 5.5.7 respectively. The former exhibited a less promising reduction in time but also constitutes a far smaller percentage of the total time t_{total} (e.g. 7.71 % versus 85.7 % for the atom move). In this section, results are presented for algorithm 14 in its entirety to ascertain whether the promising speed-up in the time taken for the triplet evaluations t_{triplet} translates into a beneficial speed-up for the full calculation.

The calculations discussed here are as outlined at the start of section 5.3. That is, in all examples, $N_{\text{a}} = 500$ and the side length L is 18 Å. A cut-off of $r_{\text{c}} = L/2 = 9$ Å is used, resulting in roughly 12 % of energy calculations being undertaken explicitly. For all cases, half of moves were accepted. Only the additive long-range correction, as shown in 2.4.4, is undertaken here, as part of the full box calculation. The GP potential was trained on a 337-point Latin hypercube, with all energies calculated at the CCSD(T) level of theory and an aug-ccPVQZ basis set. This discussion refers to the full box and atom move calculations separately, though "full box" once more refers to a full minimum image box.

Figure 5.5 shows a comparison between the speed-up achieved for the two calculations with and without PBCs and an MIC. These show that in both cases

Table 5.2: Table showing t_{total}, t_{exp} and t_{sum} for the atom move and full box calculations on a single process. The times for the former calculation are an average over 150 moves. Times in brackets are for calculations that do not include the PBCs and MICs.

Calculation	t_{total} (s)	t_{exp} (s)	t_{triplet} (s)
Atom Move	0.14 (0.88)	1.08 x 10^{-2} (1.91 x 10^{-2})	0.12 (0.77)
Full Box	21.8 (133)	2.51 (4.46)	18.8 (129)

the speed-up is reduced when the PBCs and an MIC are in use, specifically for the full box calculation. One reason for this is that, as stated, the introduction of the PBCs and MIC reduce the number of explicit energy calculations to around 12 % of the total. This results in a vast reduction in t_{triplet}, shown in table 5.2, that limits the potential gains from parallelisation.

This table illustrates that t_{total} was reduced by a factor of six for both calculations. Meanwhile, t_{exp} nearly halved and t_{triplet} was reduced by $\sim$ 84 % for both the atom move and full box calculations. Consequently, the effects of fixed costs such as instantiating arrays and sharing information between processors are exacerbated under the PBCs and MIC.

For example, in the full box calculation the time taken to set-up all arrays and share all relevant data between processors t_{set} is roughly 0.38 s, regardless of the presence of PBCs and an MIC. At $N_{\text{p}} = 40$, $t_{\text{total}} = 21.8/40 \approx 1.45$ s, meaning t_{set} is around 27 % of t_{total} for the full box calculation with these conditions. Meanwhile, without PBCs or an MIC and at the same N_{p}, $t_{\text{total}} = 133/30 \approx 4.4$ s. This means t_{set} is only 8.6 % of t_{total} in this case. The atom move calculation has fewer arrays to instantiate, hence the greater similarity between its speed-up with and without the PBCs and MIC. This is the reason that the maximum speed-up of the full box calculation no longer exceeds that of the atom move calculation in the presence of conditions, an effect that is exacerbated slightly by the marginally greater reduction in t_{sum} observed for the full box calculations.

Though the impact on the speed-up is considerable, the reduction in the amount of non-additive evaluations dramatically reduces the total time taken for both calculations under the PBCs and MIC. This can be seen by comparing

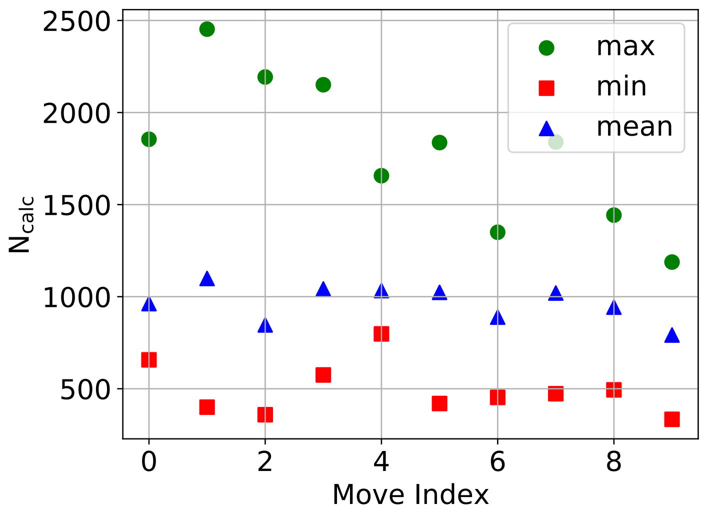


Figure 5.6: A plot showing the maximum (green), mean (blue) and minimum (red) number of non-additive calculations undertaken across 15 processors for 10 atom moves.

the bracketed values in table 5.2. These show that the times on 40 processors without the PBCs and MIC are 4.4 s for the full box and (0.88/27) x 150 $\approx$ 4.9 s for the atom move. The times that include the conditions compare favourably, with around 1.4 s each required for both calculations.

However, the increase in the proportional contribution of fixed costs does not fully explain the large reduction in speed-up observed. It must also be considered that, under an MIC, any triplets for which one or more distances exceed r_c are not calculated explicitly but handled by a long-range correction. Thus, there is a greater discrepancy in the number of calculations on each process when an MIC is used.

This is an issue because the addition of the partial sums on the root process cannot begin until all others finish determining the contributions of their triplets to the energy. Therefore, any process undertaking a disproportionately large number of explicit calculations will slow down all others. Without PBCs and an MIC this is only a concern for the atom move calculations, where the index of the moved atom affects how much work a given process had to do. This is the reason for the greater noise in the plots of the speed-up for these calculations. However, with the PBCs and MIC one process may be assigned many triplets that are calculated as part of the long-range correction, while another is assigned

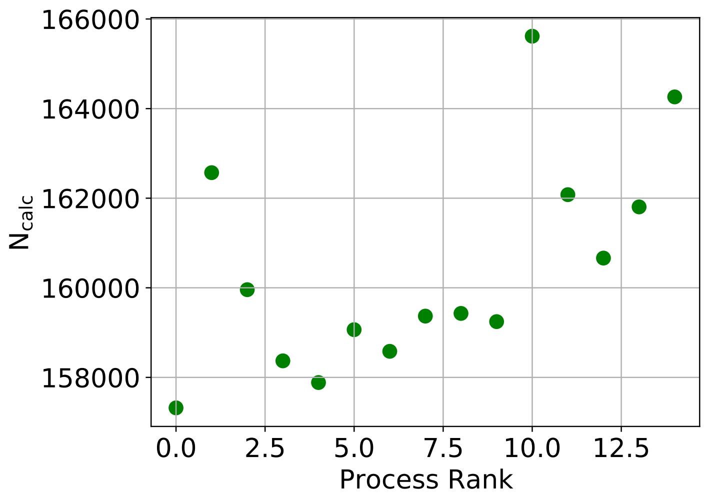


Figure 5.7: A plot showing the number of non-additive calculations undertaken across 15 processors for a full box calculation. 'Process Rank' refers to the label of each process (*i.e.* the N_{calc} value corresponding to 5.0 on the x-axis is the number of calculations on process five).

many that must be evaluated explicitly. Thus, both the full box and atom move calculations exhibit a worse speed-up with the PBCs and MIC than without.

This is evidenced in figure 5.6, which shows that for a random series of atom moves there is always at least one process that has roughly 500 more non-additive energy evaluations to undertake than average, at a minimum. Likewise, figure 5.7 shows that the least busy process has to find around 8000 fewer non-additive energies than the busiest process for a full box calculation. In terms of a percentage, the number of calculations undertaken by the least busy process for a full box calculation is now ~ 95 % of the number on the busiest. This is compared with 98.18 % with no cut-off applied (see section 5.5.4). Thus the percentage difference between the most and least busy processors has increased by a factor of ~ 2.7 %, from 1.82 % to 5 %. As these calculations are by far the most time-consuming in algorithm 14, such a disparity between the amount undertaken on each process will inevitably impact the speed-up observed on parallelisation.

The effect of this disparity is proven by examining the speed-up on the root process in: finding the non-additive energies, adding them to find the partial sum and gathering the partial sums from the other processors. That is, in the total of $t_{\text{trip}} = t_{\text{triplet}} + t_{\text{sum}} + t_{\text{gather}}$, where the latter time corresponds to the gathering

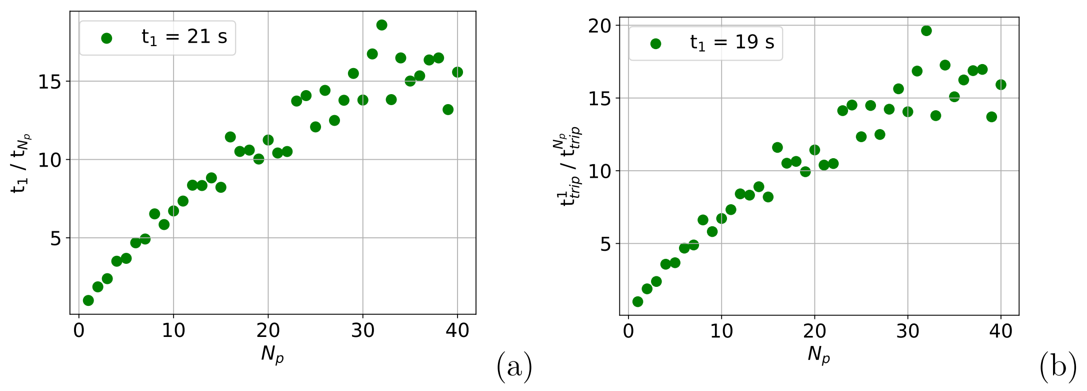


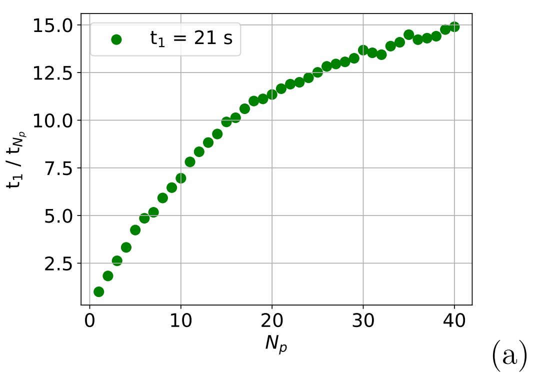
Figure 5.8: A plot showing the speed-up on the root process for the entire atom move calculation (a) and the sum of t_{triplet}, t_{sum} and t_{gather} (b). The data are averaged over 150 moves, of which half were accepted, and are over 40 processors on one node.

of the partial sums. This is because all processors will be forced to 'wait' at the final step for the busiest process to send its partial sum.

Such an examination is given in figure 5.8 for the atom move data. This figure shows the speed-up for the whole calculation, while figure 5.8(b) shows the total speed-up for t_{trip}. That they are basically identical reveals that waiting for the slow processors at the gather stage is what limits the increase in speed. This argument is reinforced when considering that the speed-up in t_{triplet} was shown to be excellent in subsection 5.5.7.

Moreover, the exponentials are only calculated for atoms at distances less than r_{c}. Thus some processors will have to calculate fewer exponentials than others. That this is not too impactful for the atom move calculation is evidenced implicitly in figure 5.8. This is because the two plots are near-identical, which implies that the speed-up of the total calculation is basically the same as that seen in t_{trip}. Thus, any change in the speed-up of the exponential calculation is effectively irrelevant. This is true of the atom move code as there are only $N_{\text{a}} - 1$ sets of exponentials to re-calculate in total. As evaluating each set is fast relative to combining them to find the non-additive energies, having some processors evaluate more exponentials than others does not have much of an impact on speed-up.

However, figure 5.8 pertains only to the atom move calculations. Figure 5.9



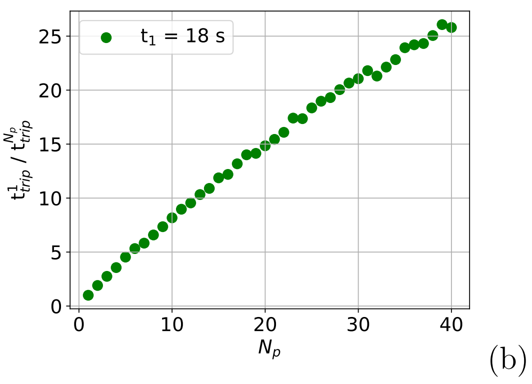


Figure 5.9: A plot showing the speed-up on the root process for the evaluation of the non-additive energy of the full box (a) and the sum of t_{triplet}, t_{sum} and t_{gather} (b). These data are over 40 processors on one node.

shows that the same conclusions cannot be drawn for the full box calculation because the speed-up in $t_{\text{trip}} = t_{\text{triplet}} + t_{\text{sum}} + t_{\text{gather}}$ exceeds the overall speed-up. Thus, other factors must cause the decrease in speed-up on introduction of the PBCs and MIC. However, the speed-up in t_{trip} is lower than that seen without any PBCs or an MIC, indicating that disparity in workload does have some small impact on speed-up.

One such contribution is the parallelisation of the exponential calculations, which was discussed in subsection 5.4.4. Though the speed-up is similar for the full box and atom move calculations, the proportion of t_{total} taken for the exponential calculations is larger for the full box calculation. Specifically, $t_{\text{exp}}/t_{\text{total}} = 2.5/21 = 12\%$ for the full box calculation on one node, versus $1.6/21 = 7.6\%$ for the atom move. Thus, when the number of processors used is large, the effect of the plateau in the exponential speed-up is greater in the full box calculation.

For example, at $N_{\text{p}} = 40$, $t_{\text{total}} = 21/15 = 1.4$ s while $t_{\text{exp}} = 2.5/11 = 0.23$ s for the full box calculation. This means that t_{exp} comprises roughly 16 % of t_{total} when $N_{\text{p}} = 40$ for this calculation. For the move calculation, meanwhile, $t_{\text{total}} = 21/15 = 1.4$ s once more but $t_{\text{exp}} = 1.5/14 = 0.11$ s. Thus t_{exp} accounts for only 7.9 % of t_{total} for the move calculation, less than half the proportion it makes up in the full box calculation. This difference is a product of the full box calculation requiring $\frac{1}{2}(N_{\text{a}}^2 - N_{\text{a}})$ sets of exponentials versus $N_{\text{a}} - 1$ for the atom move, a factor of $N_{\text{a}}/2$ more. Thus, 250 times more sets of exponentials are calculated in

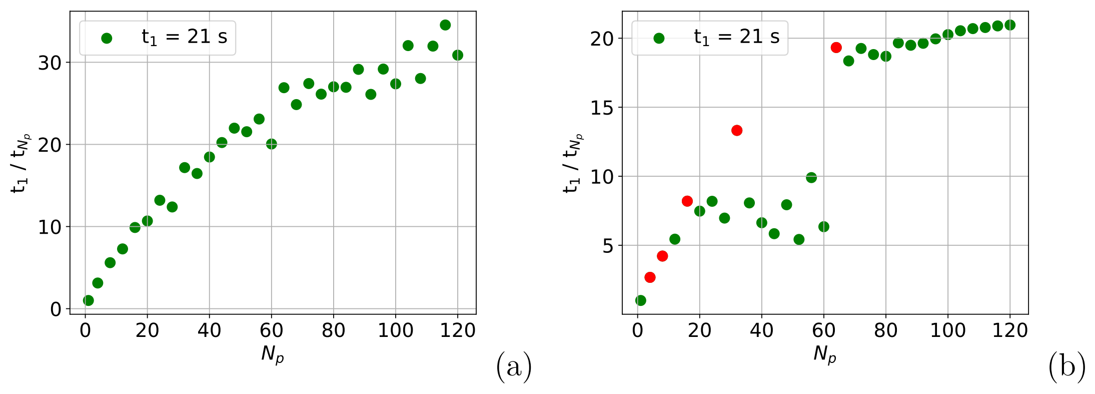


Figure 5.10: A plot showing the speed-up on the root process for the evaluation of the non-additive energy in the atom move (a) and full box (b) calculations. The data are over 120 processors across four nodes. The atom move data were averaged over 150 moves.

the full box calculation so it is more impactful when one process is allocated more pairs that require explicit calculation. The impact of this difference is reinforced when noting that each set comprises N_t x N_d exponentials.

The results presented so far show that algorithm 14 represents a promising route to implement GP potentials efficiently on a single node. However, to verify that the algorithm is general to different computing set-ups, its parallelisation across multiple nodes must be interrogated. To this end, figure 5.10 shows the results for the atom move and full box calculations across 120 processors split evenly over four nodes. Just as in subsection 5.5.7, the even distribution means that, for example, when $N_\mathrm{p} = 120$, there are $120/4 = 30$ processors on each node.

For the atom move calculation, shown in figure 5.10(a), the speed-up roughly doubles across the further 80 processors when compared to the data in figure 5.8. This evidences that this calculation parallelises well over processors distributed across multiple nodes, which is a promising result given displacement calculations are the most common in Monte Carlo simulations. These results correspond to a time of two thirds of a second for the 150 evaluations of ΔU_NA.

The full box calculation, displayed in figure 5.10(b), exhibits less promising results. The total speed-up increases from 15-fold over 40 processors to 20-fold over 120 processors only. Moreover, between $N_\mathrm{p} = 20$ and $N_\mathrm{p} = 60$ no consistent increase in speed-up is seen.

Part of the reason for the lower overall speed-up will be the parallelisation of t_{exp}. It has already been shown that this is a greater proportion of t_{total} for the full box calculation and the way in which the processors are divided across the nodes means the maximum benefit from parallelisation of t_{exp} is not gained until $N_{\mathrm{p}} = 80$. For example, on one process $t_{\mathrm{exp}} = 2.45$ s, meaning on 20 processors over four nodes $t_{\mathrm{exp}} = 2.45/4 \approx 0.61$ s. Meanwhile $t_{\mathrm{total}} = 21/7.5 = 2.8$ s, meaning t_{exp} is roughly 22 % of t_{total} for the full box calculation. Compare this with the calculation on one node, where $t_{\mathrm{exp}} = 2.45/13 \approx 0.19$ s on 20 processors. As $t_{\mathrm{total}} = 20/11 \approx 1.8$ s at this N_{p} on a single node, t_{exp} was only around 11 % of t_{total}.

The failure of the speed-up to increase between $N_{\mathrm{p}} = 20$ and $N_{\mathrm{p}} = 60$ cannot be explained in this way. Nor is it rationalised with the distribution of triplets, as it begins at a value of N_{p} that was evaluated on one node. However, a trend akin to that in figure 5.9(a) is observed in figure 5.10(b) over the red points on the latter. Each red point represents a calculation for which the total number of processors was a power of two (*i.e.* $N_{\mathrm{p}} = 4$ for 2^2, $N_{\mathrm{p}} = 8$ for 2^3, *etc.*). At these points the speed-up follows the same pattern as on a single node. For example, a 13-fold speed-up is observed for a 32-process calculation whether on one or four nodes. This suggests that the anomalous results, which are reproducible with some degree of noise, are a product of the HPC architecture rather than a coding concern. Furthermore, the plateau at a 20-fold speed-up seen in figure 5.10(b) seems in agreement with the decline in speed-up observed in figure 5.9(a). This suggests further that distribution over multiple nodes does not affect the performance of the algorithm.

Thus algorithm 14 represents an efficient method for parallelising calculations with GP potentials over processors on one or many nodes, with the only results indicative of poor parallelisation not a consequence of the algorithm itself. However, it must be verified that the speed-up reaches a plateau only when the fixed costs dominate the calculation time.

This is confirmed by figure 5.11, which shows the decline in calculation time as N_{p} increases alongside the times of the fixed costs for both calculations. This figure highlights that for $N_{\mathrm{p}} > 60$, t_{total} is dominated by the fixed costs in both

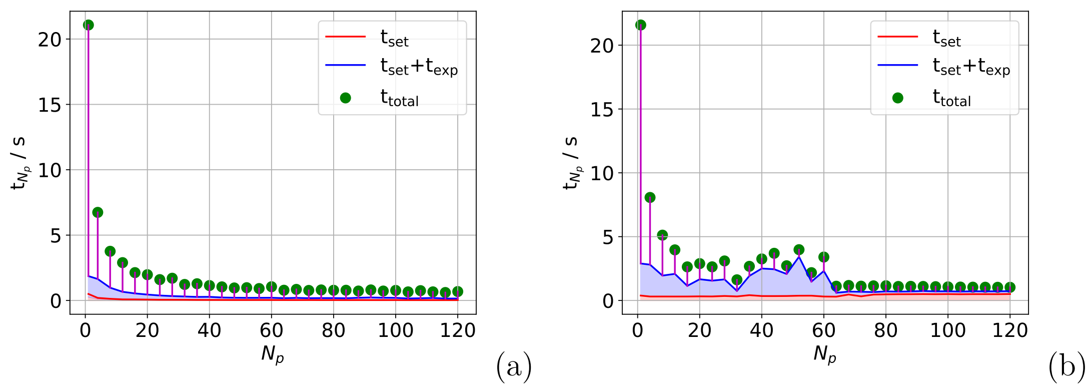


Figure 5.11: A plot of t_{total} against N_{p} on the root process for the evaluation of the non-additive energy in the atom move (a) and full box (b) calculations. The horizontal red and blue lines indicate the contributions of the set-up and exponential calculations respectively, while the vertical purple lines show the value of t_{trip} for each calculation.

calculations. Thus any further efficiency gains from the method would require reductions in these fixed costs, rather than improvements to the parallelisation. This figure also displays the aforementioned difference in the fixed costs of the two calculations. In addition, figure 5.11 shows that t_{exp} is erratic for the full box calculation. Once again, such a result is a consequence of the architecture of the HPC, which is exacerbated by the discrepancy in the number of exponential evaluations undertaken on each process. Recall that this discrepancy is greater for the full box calculation.

This figure also implies that a more pronounced speed-up on parallelisation would be observed for higher-dimensional systems. For systems including different atomic and molecular species, a larger training set would be required, which would result in a larger cost of prediction. Consequently, the cost of evaluating the triplet energies, the largest contribution to t_{total}, would be larger. As this evaluation parallelises well, such a change would result in a plateau at higher N_{p}. This is attainable with the methods presented here, given that the algorithm parallelises well across multiple nodes assuming all processors will be used on each.

Furthermore, the parallel speed-up will be improved if the number of atoms in the box increases. This is because the number of triplets $N_{\text{tri}} \propto N_{\text{a}}^3$ while the total number of exponentials $N_{\mathrm{e}} \propto N_{\mathrm{a}}^2$. Therefore the triplet summation, which parallelises near-perfectly, becomes a proportionally larger contribution to t_{total}. Moreover, the number of atoms within r_{c} of some atom α, N_{a}^{α}, will be proportional to the volume of the cut-off sphere. That is, $N_{\mathrm{a}}^{\alpha} \propto r_{\mathrm{c}}^3$. Therefore, at a fixed density, as r_{c} increases the number of explicit triplet calculations around α, $N_{\mathrm{tri}}^{\alpha}$, will increase as $N_{\mathrm{tri}}^{\alpha} \propto (N_{\mathrm{a}}^{\alpha})^3 \propto r_{\mathrm{c}}^9$. Given that $N_{\mathrm{e}}^{\alpha} \propto N_{\mathrm{a}}^2 \propto r_{\mathrm{c}}^6$, $N_{\mathrm{tri}}^{\alpha}/N_{\mathrm{e}}^{\alpha} \propto r_{\mathrm{c}}^3$. This shows the portion of the calculation that parallelises excellently will increase rapidly with the cut-off, which will grow larger when box size is increased. Thus the efficiency of the parallelisation will be enhanced greatly by both increasing the density or size of the simulation box. This implies that for more complex applications, such as for larger molecules or systems closer to the critical point, the algorithm will exhibit even better speed-up.

5.9 Conclusions

The results in section 5.8 show that the atom move and full box calculations parallelise well for a single node when PBCs and an MIC are applied. This parallelisation corresponds to a speed-up of a factor of around 15 in both calculations, meaning 150 evaluations of ΔU_{NA} and one evaluation of U_{NA} take around 1.4 s each. Moreover, it has been shown that the calculation of exponentials has little impact on the overall speed-up, while the combination of exponentials to determine energies parallelises well.

Both calculations exhibit markedly lower speed-up in the presence of the PBCs and MIC. For the atom move calculation this is fully explained by the discrepancy in workload brought about by energies and exponentials being set to zero for interactions where $r > r_{\mathrm{c}}$. The full box calculation is also affected by this discrepancy, but to a lesser extent. The reduction in speed-up for this calculation requires additional consideration of the fixed costs. These are far greater for the full box calculation, as is the contribution of t_{exp}, even before it plateaus at $N_{\mathrm{p}} > 19$. These costs are more significant in the presence of the MIC as the time needed to assess the triplet energies, which parallelises excellently, is reduced greatly.

When calculations are spread over 120 processors on four nodes, the atom move calculations undergo a significant speed-up. Between $N_\mathrm{p} = 40$ and $N_\mathrm{p} = 120$, the speed-up for the calculation doubles. The rate of speed-up, however, is lower than observed on a single node. This is unsurprising as the speed-up per process on a single node begins to slow as N_p approaches 40. Thus the slower speed-up above this N_p should be lower, as fixed costs become more significant. That distributing the calculation over multiple nodes does not affect performance is further confirmed because the speed-up on an equivalent number of processors is similar regardless of whether the processors are on a single node or four.

Under the same conditions, the full box calculation exhibits a smaller increase in speed-up, from 15-fold to 20-fold over the extra 80 processors. This, however, seems to fit with the trend seen on a single process, where the speed-up is beginning to plateau as N_p approaches 40. Consequently, it is not indicative of poor speed-up in this calculation over many nodes. Instead, it is a product of the aforementioned increase in the significance of the fixed costs for this calculation in the presence of the MIC. This is highlighted by the fact the full box calculation parallelises more effectively than the atom move on one node when no PBCs or MIC are used.

The full box calculation also shows a region where the speed-up fails to increase is seen between $N_\mathrm{p} = 20$ and $N_\mathrm{p} = 60$. That this is not the case for calculations that are across a number of processors that is a power of two (*i.e.* for which $N_\mathrm{p} = 2^x$) suggests that this is an issue with the HPC architecture, whereby nodes that are in clusters of pairs can communicate more quickly. In fact, the calculations with $N_\mathrm{p} = 2^x$ match the speed-up observed on a single node.

Furthermore, both calculations only plateau as the fixed costs come to dominate the calculation time. Any improvement in efficiency would therefore be gained from reducing the fixed costs, rather than streamlining the algorithm further. The only clear area of improvement outside of the fixed costs is in reducing the discrepancy in workload across processors. A method for this is described in the future work.

The main drawback of the data presented is that additive calculations are not included in the benchmarking. However, the algorithms for these calculations are simpler. That is, they involve fewer exponentials and the number of atom pairs in the simulation will be fewer than the number of triplets provided $\frac{1}{6}(N_{\mathrm{a}}^3 - 3N_{\mathrm{a}}^2 + 2N_{\mathrm{a}}) > \frac{1}{2}(N_{\mathrm{a}}^2 - N_{\mathrm{a}})$. This is the case as long as $6N_{\mathrm{a}}^2 < N_{\mathrm{a}}^3 + 5N_{\mathrm{a}}$, which holds for $N_{\mathrm{a}} > 5$. This value will be exceeded in any molecular simulation, especially any concerned with the phase boundaries of fluids such as those that could be employed for carbon capture and storage pipelines. Combined with the fact that all methods for implementing additive calculations are simplifications of their non-additive equivalents, it is not anticipated that their addition will have a negative effect on speed-up.

Thus algorithm 14 represents an efficient method for evaluating the non-additive energy of a simulation with a GP potential. This implementation reduces calculation time for both atom move and full box calculations significantly. Such a result is especially promising for the atom move calculation, as this is undertaken most frequently in any Monte Carlo simulation. That the algorithm only plateaus as the fixed costs become relatively large suggests that further improvement will not be easy, while its ability to run on one or many nodes makes it robust to different computational set-ups. Finally, it is likely that the speed-up presented here would be increased if the same algorithm were applied to a larger box, implying its effectiveness in simulations of systems closer to the critical point, for example, would be better still.

5.10 Future work

Though the results presented in section 5.8 are of great promise, modifications must be made to produce an algorithm that is suitable for a proof-of-concept simulation of the phase coexistence of liquid and gaseous argon with a GP potential. Namely these are additive energy calculations (discussed throughout sections 5.3.1 to 5.6), a short-range cut-off for the additive and non-additive energies, particle exchange moves and three-body long-range corrections. Though the latter is often set to zero, a potential correction is presented here nonetheless. In addition, an overview of a method to reduce the disparity in workload between different processors under the PBCs and MIC is given. All of these changes are discussed in subsection 5.10.1.

Beyond the proof-of-concept simulation, the method must generalise to Monte Carlo simulations of arbitrary species. This is required to model phase coexistence in, for example, carbon capture and storage pipelines. The requisite modifications for this are given in subsection 5.10.2. This overview begins with an extension to mixtures of atomic species, before detailing how to include molecules.

Finally, to evaluate non-equilibrium properties such as viscosity in molecular mixtures, Monte Carlo simulations are not viable. Therefore an extension of the method to molecular dynamics is needed, which requires force calculations. Fortunately, for a kernel that can be differentiated the forces can be obtained straightforwardly. This is discussed in subsection 5.10.3.

5.10.1 Modifications for a Monte Carlo simulation of Ar

Short-range cut-offs

As the simplifications required to determine the additive energy were discussed throughout this chapter, the first modification to algorithm 14 given here is the short-range cut-off. This is a simple extension to the method already presented. The non-additive interactions require simply that the energy is set to zero for all triplets containing particles separated by less than a certain distance R_{short}. A sensible choice would be $R_{\text{short}} = R_{\text{min}}$, the minimum distance in the reference LHC used to train the GP.

For additive energy calculations, energies of configurations at separations below R_{short} are approximated by a simple short-range function. An appropriate function is any for which $U \to \infty$ as $r \to 0$, such as that suggested by Uteva *et al.* [27]

$$U = U_{\text{max}} \frac{1}{N} \sum_{i=1}^{N} (x_i / x_{\text{max}})^{12}. \tag{5.21}$$

In this equation, $x_{\text{max}} = R_{\text{min}}^{-1}$, x_i is an inverse interatomic distance, U_{max} is the energy at x_{max} and N is the number of configurations in the training set. The exact form of the function is not overly important because any moves that bring particles within R_{short} of each other will result in an increase in energy sufficient to make rejection probable.

Particle exchange moves

The capacity to undertake particle exchange moves is key to simulations of phase coexistence using the Gibbs ensemble (see Chapter 2.4). Consequently, this capability must be added to algorithm 14 for any proof-of-concept argon simulation of this property.

For insertion of a particle into the gas phase from the liquid phase this is straightforward. A random particle from the latter must be selected and removed, with the energy of the liquid phase re-calculated thereafter. This calculation requires only the deduction of the interaction energy of this particle with all others from the total potential of the atoms in the liquid. Likewise, the new gas-phase energy must be found, which entails determining the interactions of the newly-introduced atom with all others. This can be done similarly to finding ΔU_{NA} or ΔU_{Add}. If the energy of these interactions is negative, the move is accepted. Otherwise, it is either accepted or rejected as outlined in Chapter 2.4.

Insertion into the liquid can prove more troublesome. Although the requisite energy calculations are identical, the probability of rejection is far higher. This is due to the relatively dense packing of the particles in the liquid phase, which makes it likely that the new particle will be inserted into an area where it undergoes strong repulsive interactions. To circumvent this, the particle is often introduced with its interactions 'turned off' [148]. They are re-introduced slowly over a series of atom displacement moves that leave the newly-introduced particle in a position where its potential is not dominated by repulsive interactions.

Non-additive long-range corrections

Non-additive long-range corrections are more treacherous to apply than their additive counterparts. This is because the introduction of a third atom creates different scenarios, each of which has a unique correction. These scenarios must, therefore, be distinguished prior to the application of any correction.

Throughout this discussion, the atoms that comprise the triplet are labelled 1, 2 and 3 such that $r_{12} < r_{13} < r_{23}$. There are three different scenarios that result, each of which requires a unique long-range correction:

- $r_{\mathrm{c}} < r_{12} < r_{13} < r_{23}$, denoted Type 1.

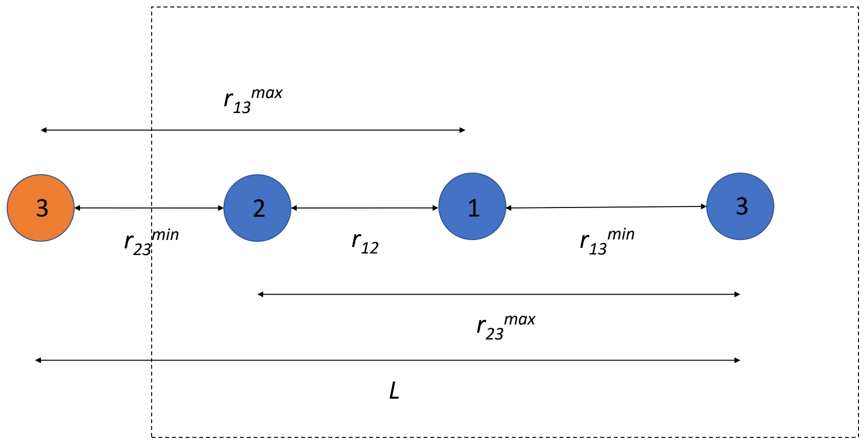


Figure 5.12: A Type 3 triplet where $r_{23}^{\text{max}} > L/2$, meaning it must exceed r_c. The dashed line is the MIC box centred on atom one and the orange atom is a periodic image of atom three.

- $r_{12} < r_\text{c} < r_{13} < r_{23}$, denoted Type 2.
- $r_{12} < r_{13} < r_\text{c} < r_{23}$, denoted Type 3.

Because all interacting species are atomic they can always be rotated into the same plane, though the following method could be extended to molecules regardless.

Type 3

Considering first the Type 3 scenario, figure 5.12 reveals that selecting which triplet to evaluate is complicated in this scenario. In this example, taking $r_{23} = r_{23}^{\text{min}}$ would redefine atom 2 as atom 1. Meanwhile, the previous atom 1 would become atom 2. This too is a viable triplet, the energy of which must be accounted for.

This problem is solved by a sensible choice of L. The figure shows that $L \geq r_{12} + r_{13}^{\text{min}} + r_{23}^{\text{min}}$. In the case where the separations are as large as possible while still defining the Type 3 scenario, $r_{12} = r_{13}^{\text{min}} = r_{23}^{\text{min}} = r_\text{c}$. Thus, $L \geq 3r_\text{c}$ so a side length greater than three times the cut-off distance will avoid the issue of how to define the triplets.

The region of integration for the Type 3 correction is also not straightforward to define as it covers a region outside of the cut-off sphere around atom 2 but within that of atom 1. This issue can be solved by adding an additional specification to the LHC used to train the GP. Including points where $r_{23}^{\max} = r_{13}^{\max} + r_{12}$ in the training LHC will allow the GP to predict explicitly the energies for this scenario. Thus no long-range correction for the Type 3 scenario is required.

Type 2

Under the Type 2 scenario, meanwhile, the long-range correction accounts for the effect of 3 on the 1-2 interaction when the former is at long-range. Thus the Type 2 long-range correction $U_{\mathrm{T2}}^{\mathrm{LRC}}$ can be expressed as an integral over the position of atom 3 only. This yields

$$U_{\mathrm{T2}}^{\mathrm{LRC}} = \int E^{(3)} d\mathbf{r}_3, \tag{5.22}$$

where $\mathbf{r}_3$ are the Cartesian coordinates of atom 3 and $E^{(3)}$ is an expression for the non-additive energy at long-range. The latter is discussed later in this section.

However, all expressions for the energy discussed throughout this work have been given in interatomic distances. An integral of an equivalent format will be over both r_{13} and r_{23}. As $r_{13} < r_{23}$, atom 3 must be within the hemisphere around atom 1 that leaves it closer to 1 than 2. r_{13} must therefore be between the cut-off and infinity, while r_{23} must exceed r_{13} and be less than $r_{12} + r_{13}$ to obey the triangle rule. The integral therefore becomes proportional to

$$U_{\mathrm{T2}}^{\mathrm{LRC}} \propto r_{12} \int_{r_{\mathrm{c}}}^{\infty} \int_{r_{13}}^{r_{12}+r_{13}} E^{(3)} r_{13} r_{23} dr_{23} dr_{13}. \tag{5.23}$$

Figure 5.13 shows that these distances can be re-written in terms of the angle θ and distance x using the law of cosines. When doing so, the full integral can be written as

$$U_{\mathrm{T2}}^{\mathrm{LRC}}(r_{12}) = 4\pi N_{\mathrm{a}} \rho \int_{0}^{\frac{\pi}{2}} \int_{x_{\min}}^{\infty} x^2 \sin\theta E^{(3)} dx d\theta. \tag{5.24}$$

The limits on the integral over θ are a product atom 3 being closer to atom 1, which is also the reason the constant on the front is 4π rather than 2π. Meanwhile, $x_{\min}$ is the value of x at which $r_{23} = r_{\mathrm{c}}$ and ρ is the density of atoms in the simulation box. The dependence of the Type 2 correction on r_{12} stems from these two atoms being within r_{c} of each other. Thus this integral could be evaluated for different values of r_{12} and stored in a look-up table, which would have to be

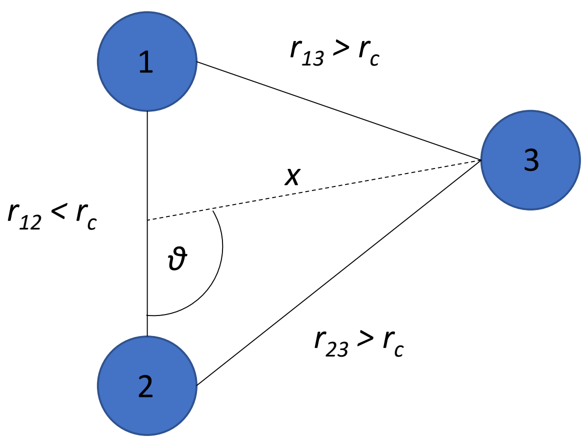


Figure 5.13: A sketch of a Type 2 triplet that shows the angle θ between r_{12} and the distance x.

updated following a volume change. The correction for any Type 2 triplet could then be evaluated by interpolation.

Type 1

The Type 1 scenario, meanwhile, is similar to Type 2 as illustrated in figure 5.13, with the single difference being that $r_\mathrm{c} < r_{12}$ in the former. An equivalent to equation (5.24) can therefore be written down for the Type 1 correction. However, because r_{12} is now greater than the cut-off, this distance must also be integrated over. Thus the integral for the Type 1 long-range correction is

$$U_{\mathrm{T1}}^{\mathrm{LRC}} = 8\pi^2 \frac{N_\mathrm{a}\rho^2}{3!} \int_{r_\mathrm{c}}^{\infty} \int_0^{\frac{\pi}{2}} \int_{x_\mathrm{min}}^{\infty} r_{12}^2 x^2 \sin\theta E^{(3)} dx d\theta dr_{12}. \tag{5.25}$$

This integral is similar to the two-body correction discussed in Section 2.4.4. That is, because all distances are in excess of r_c it is a purely mean field correction. Consequently, this integral need only be evaluated at the outset of the simulation and whenever the volume changes. The integrals in equations (5.24) and (5.25) represent a more general approach to solving the long-range correction for three-body interactions than that proposed by Leonhard and Deiters [131] as they can employ any desired function for $E^{(3)}$ and do not disregard any scenario that requires a correction.

Form for $E^{(3)}$

The three-body energy $E^{(3)}$ in the above integrals can be estimated as

$$E^{(3)} = \frac{A(\theta)}{r_{12}^3 r_{13}^3 r_{23}^3}, \tag{5.26}$$

where $A(\theta)$ is a constant for a given value of θ. $A(\theta)$ is obtained by estimating the energy with a GP when all distances are scaled such that the largest is no greater than r_c.

For a Type 2 triplet such as that in figure 5.13, this scaling is achieved by moving atom 3 along x until $r_{13} < r_{23} \leq r_\mathrm{c}$. The new values of the distances that satisfy this inequality are referred to as r'_{13} and r'_{23}. As no distances now exceed r_c, the energy of the scaled triplet can be estimated with a GP,

$$E^{(3)}_\mathrm{GP} = \frac{A(\theta)}{r_{12}^3 (r'_{13})^3 (r'_{23})^3} \leftrightarrow A(\theta) = E^{(3)}_\mathrm{GP} r_{12}^3 (r'_{13})^3 (r'_{23})^3. \tag{5.27}$$

It follows that

$$E^{(3)} = E^{(3)}_\mathrm{GP} \left(\frac{r'_{13}}{r_{13}}\right)^3 \left(\frac{r'_{23}}{r_{23}}\right)^3. \tag{5.28}$$

For a Type 1 triplet, a similar approach can be adopted. However, rather than simply sliding atom three along x, the first step is to scale all distances down such that $r_{12} < r_\mathrm{c}$. Then the atom is moved along x until $r_{13} < r_\mathrm{c}$ before a final scaling of all distances to ensure that $r_{23} < r_\mathrm{c}$. Denoting the scaled r_{12} as r'_{12}, this leads to

$$A(\theta) = E^{(3)}_\mathrm{GP} (r'_{12})^3 (r'_{13})^3 (r'_{23})^3 \tag{5.29}$$

and

$$E^{(3)} = E^{(3)}_\mathrm{GP} \left(\frac{r'_{12}}{r_{12}}\right)^3 \left(\frac{r'_{13}}{r_{13}}\right)^3 \left(\frac{r'_{23}}{r_{23}}\right)^3. \tag{5.30}$$

Equations (5.28) and (5.30) can be substituted respectively into equations (5.24) and (5.25). Before doing this, however, r_{13} and r_{23} must be expressed in terms of the integration variables x, r_{12} and θ. These expressions can be derived straightforwardly from the triangle rule, with

$$r_{13} = \left(\frac{1}{4} r_{12}^2 + x^2 + r_{12} x \cos\theta\right)^{\frac{1}{2}} \tag{5.31}$$

and

$$r_{23} = \left(\frac{1}{4} r_{12}^2 + x^2 - r_{12} x \cos\theta\right)^{\frac{1}{2}}. \tag{5.32}$$

Method to reduce disparity in triplet calculations

It was established in section 5.8 that the uneven distribution of the triplets that need to be calculated explicitly caused a reduction in speed-up under the PBCs and MIC. The issue could be alleviated by ensuring the discrepancy between the number of evaluations on a process and the average across all processors is reduced.

Such a reduction could be achieved with a few simple steps. First, all processors must calculate the average number of calculations N_{av} across all processors. Then, each process evaluates how many it must undertake, N_{self}. This would require only a simple extension of the pre-existing code used to determine whether or not to find the energy explicitly. Thereafter all processors that have $N_{\mathrm{self}} > N_{\mathrm{av}} + cN_{\mathrm{av}}$, where $c \in (0, 1]$, send cN_{av} jobs to another process that has $N_{\mathrm{self}} + cN_{\mathrm{av}} < N_{\mathrm{av}}$.

This fix is general to both atom move and full box calculations. However, it is only feasible if cN_{av} is small enough that sharing the triplets does not come to dominate the calculation time. This could be circumvented by having all processors share a matrix of triplets. Then processors that need to reduce their workload could share the indices of rows in that matrix with processors that have fewer calculations to undertake.

5.10.2 Extension to atomic and molecular mixtures

Extension to atomic mixtures

When considering systems comprising multiple atomic species, the energy of all triplets can still be evaluated using equation (5.8). However l_j now varies with j, as the atom types participating in the interactions described by each element in $\mathbf{x}$ may differ. To determine the appropriate l_j to use for a given interaction, knowledge of the species undergoing the interaction must be employed.

To this end a vector of atom types $\mathbf{A}$, which contains the species of each atom in the cluster, of length N_{a} is introduced. Thereafter an operator $\hat{f}(\alpha, \beta)$ is used, where for any α, β pair within a triplet

$$\hat{f}(\alpha, \beta) = l_{\alpha,\beta}$$

That is, for a given pair of atoms within a triplet, $\hat{f}(\alpha, \beta)$ uses $\mathbf{A}(\alpha)$ and $\mathbf{A}(\beta)$ (*i.e.* the atoms that comprise the pair) to return an appropriate lengthscale.

Another problem introduced by the inclusion of multiple atomic species is the choice of permutation matrix $\overline{\mathbf{P}}$. For example, consider a simulation comprising Ne and Ar atoms. Any triplet therein could have one of four compositions: 3Ar, 2Ar-Ne, 2Ne-Ar, 3Ne. Though the first and last compositions would share the same $\overline{\mathbf{P}}$ (that shown in subsection 5.5.1), the two compositions that contain both atom types have permutation matrices that vary based on the ordering of the atoms. To illustrate this, consider a single 2Ar-Ne triplet, for which (Ne, Ar, Ar). For such a triplet

$$\overline{\mathbf{P}} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}.$$

(Recall that it is not the atoms being switched but the interatomic distances.) However, it is equally reasonable to write (Ar, Ar, Ne) and hence

$$\overline{\mathbf{P}} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}.$$

This problem is easily fixable by associating with each atom type a weight value, w_{atom}. These could be stored in a vector $\mathbf{w}$ that contains in its ith element the, for example, atomic number of atom i. Such an approach permits triplets with the same atomic composition to be denoted consistently. In the given example of $\mathbf{w}$, $w_{\text{Ar}} > w_{\text{Ne}}$ and all triplets with the 2Ar-Ne composition would be written (Ar, Ar, Ne). Therefore, these triplets would use the second $\overline{\mathbf{P}}$ shown above. Similarly, such a weighting system ensures all 2Ne-Ar triplets are (Ar, Ne, Ne) and use the first $\overline{\mathbf{P}}$ shown above. Consequently, each triplet composition will have a well-defined permutation matrix.

The above method extends to systems of more than two atomic species, with energy calculations still possible using equation (5.8). The consideration of the lengthscales affects the exponential calculations only; the combination of exponentials to find the non-additive/additive energies will still proceed as before once an appropriate permutation matrix has been found. Thus, the necessary changes to the current algorithm are minor. Recovery of the appropriate lengthscale before the exponential calculations and determination of $\overline{\mathrm{P}}$ before the energies are evaluated are the only additional steps needed.

Mixtures containing molecular species

To extend the methodology to molecular species requires $\mathbf{A}$ to be extended to a N_{a} x N_{max} matrix $\overline{\mathbf{A}}$, where N_{max} is the number of atoms present in the largest molecule and N_{a} now refers to the number of particles. The form of $\overline{\mathbf{A}}$ can be illustrated by the simple example of a 2Ar-CO triplet, for which

$$\overline{\mathbf{A}} = \begin{pmatrix} \mathrm{C} & \mathrm{O} \\ \mathrm{Ar} & \mathrm{ND} \\ \mathrm{Ar} & \mathrm{ND} \end{pmatrix}$$

and “ND” stands for “not defined”.

The operator $\hat{f}(\alpha, \beta)$ would then have to be extended to three dimensions, becoming $\hat{f}(\alpha, \beta, j)$. While α and β still refer to particles in the triplet, j refers to the jth interaction between these particles. For example, $\hat{f}(1, 2, 2)$ would return the lengthscale for the second interatomic distance between the CO and the first Ar. If $\hat{f}(2, 3, 2)$ were picked, the exponential calculation would be skipped as the Ar-Ar interaction is one-dimensional.

$\overline{\mathbf{P}}$ could once again be determined with a vector of weights $\mathbf{w}$, where this time the ith entry therein would be the sum of the atomic numbers in the $i$the row of $\overline{\mathbf{A}}$. For the 2Ar-CO interaction, $w_{\mathrm{Ar}} > w_{\mathrm{CO}}$. This means that distance 1 is $r_{\mathrm{Ar-Ar}}$. As finding w_{CO} requires the atomic numbers of C and O, one can verify that $w_{\mathrm{O}} > w_{\mathrm{C}}$. Thus, distances 2 and 3 are O-Ne distances, while 4 and 5 are C-Ne distances. This yields

$$\overline{\mathbf{P}} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 2 & 4 & 5 \\ 1 & 3 & 2 & 5 & 4 \\ 1 & 2 & 3 & 5 & 4 \end{pmatrix}.$$

Once more, this method extends to different mixtures of molecular and atomic species. Modifications to the existing code to run Monte Carlo simulations of phase coexistence should therefore be relatively straightforward, requiring only some additional steps in algorithm 14 rather than a totally new algorithm.

5.10.3 Forces for molecular dynamics simulations

Any molecular dynamics simulation requires the forces acting on each atom. A useful property of GP potentials in this context is that forces can be obtained by differentiating the kernel function. That is, a separate GP does not need to be trained on force data.

For argon atoms, forces can be found from differentiating equation (5.5). For an atom δ, forces must be found in each of the x, y and z directions. Letting d denote an inverse interatomic distance for clarity, the force on atom δ from direction λ is

$$F_\lambda = -\sigma_f^2 \sum_{j\neq\delta}^{N_\mathrm{a}} \sum_{i=1}^{N_\mathrm{t}} \Lambda_i \frac{\partial}{\partial \lambda_\delta} \left[\exp\left(-\frac{(d_{\delta j} - d_i)^2}{2l^2} \right) \right]. \tag{5.33}$$

The sum over all atoms represents the force being an accumulation of the forces exerted by all other atoms along a given axis.

As mentioned, d is an inverse distance, so

$$d_{\delta j} = \frac{1}{[(\mathrm{x}_\delta - \mathrm{x}_j)^2 + (\mathrm{y}_\delta - \mathrm{y}_j)^2 + (\mathrm{z}_\delta - \mathrm{z}_j)^2]^{1/2}} \tag{5.34}$$

and the substitution

$$v = \frac{1}{[(\mathrm{x}_\delta - \mathrm{x}_j)^2 + (\mathrm{y}_\delta - \mathrm{y}_j)^2 + (\mathrm{z}_\delta - \mathrm{z}_j)^2]^{1/2}} - d_i \tag{5.35}$$

can be used. This allows one to solve the derivative in equation (5.33) using the chain rule. Writing $k = \exp\left(-\frac{(d_{\delta j}-d_i)^2}{2l^2}\right) = \exp\left(-\frac{v^2}{2l^2}\right)$,

$$\frac{\partial k}{\partial \lambda_\delta} = \frac{\partial k}{\partial v}\frac{\partial v}{\partial \lambda_\delta}. \tag{5.36}$$

Letting $q = (\mathrm{x}_\delta - \mathrm{x}_j)^2 + (\mathrm{y}_\delta - \mathrm{y}_j)^2 + (\mathrm{z}_\delta - \mathrm{z}_j)^2$, one has that $v = q^{-1/2} - d_i$. Thus from

$$\frac{\partial v}{\partial \lambda_\delta} = \frac{\partial v}{\partial q}\frac{\partial q}{\partial \lambda_\delta}, \tag{5.37}$$

one obtains

$$\begin{aligned} \frac{\partial v}{\partial \lambda_\delta} &= -\frac{1}{2}\frac{2(\lambda_\delta - \lambda_i)}{[(\mathrm{x}_\delta - \mathrm{x}_j)^2 + (\mathrm{y}_\delta - \mathrm{y}_j)^2 + (\mathrm{z}_\delta - \mathrm{z}_j)^2]^{3/2}} \\ &= -(\lambda_\delta - \lambda_i) d_{\delta j}^3. \end{aligned} \tag{5.38}$$

Taking

$$\frac{\partial k}{\partial v} = -\frac{v}{l^2}\exp\left(-\frac{v^2}{2l^2}\right) = -\frac{(d_{\delta j} - d_i)}{l^2}\exp\left(-\frac{(d_{\delta j} - d_i)^2}{2l^2}\right) \tag{5.39}$$

and substituting equation (5.38) into equation (5.36) gives

$$\frac{\partial k}{\partial \lambda_\delta} = \frac{d_{\delta j}^3 (d_{\delta j} - d_i)(\lambda_\delta - \lambda_i)}{l^2} \exp\left(-\frac{(d_{\delta j} - d_i)^2}{2l^2}\right). \tag{5.40}$$

This gives the force along direction λ as

$$F_\lambda = -\sigma_f^2 \sum_{j \neq \delta}^{N_\mathrm{a}} \sum_{i=1}^{N_\mathrm{t}} \Lambda_i \frac{d_{\delta j}^3 (d_{\delta j} - d_i)(\lambda_\delta - \lambda_i)}{l^2} \exp\left(-\frac{(d_{\delta j} - d_i)^2}{2l^2}\right). \tag{5.41}$$

The form of (5.41) implies that the same strategy used to calculate the energy can be applied to evaluating forces. The exponentials can still be pre-calculated and combined in the same way, only the latter action requires the additional evaluation of the two fractions within the sum. This should be computationally insignificant and the method can be generalised to mixtures of atoms and molecules.

Chapter 6

Final discussion

It has been mentioned many times throughout the antecedent chapters that Gaussian process (GP) potentials offer the best path to the application of high-level *ab initio* information in molecular simulations of fluids comprising small molecules. This is because GPs are capable of accurate interpolation of multidimensional functions and require a single, general algorithm to do so. That is, the method used to train a GP potential is not affected by the potential itself; all that is needed is relevant training data. Furthermore, GPs can learn a potential energy surface with fewer training points than equivalent methods, which renders them ideal for applying, for example, coupled cluster-level potentials via transfer learning. They are therefore excellent for use in simulations of liquids, for which equations of state cannot be obtained from the virial expansion. However, the capacity of GPs to facilitate first principles, quantitatively accurate simulations is inhibited by their computational cost, which is high compared to other statistical methods and, in particular, traditional force fields. Thus it is paramount when attempting to obtain thermophysical properties of liquids from first principles to train and implement GP potentials efficiently.

A more efficient training method was discussed in Chapter 3, which demonstrated that GP potentials can be developed with significantly fewer training points by varying the cross-over distance between the GP itself and a long-range function. The long-range function was relatively simple, with a closed functional form that ensures it was computationally inexpensive to evaluate. The cross-over distance was varied using boundary optimisation, a novel method that learned the optimal cross-over distance from the training data and permitted the imposition of different cross-over distances on different interatomic interactions. This method was demonstrated to be successful for a variety of linear intermolecular systems encompassing a multitude of interaction types, and addressed complications relating to hyperparameter stability in training of $(HF)_2$ and $(HCl)_2$ potentials. It was found that for non-$(HX)_2$ systems, boundary optimisation reduced the number of training points at fixed, usable accuracy by up to 40 %, with a typical reduction of 15-30 %. Furthermore, boundary optimisation in one or many dimensions incurred an increase in training time so negligible as to be unnoticeable, and is general to other methods of prediction. For example, GP predictions could be replaced by those of a neural network if desired. The technique also addressed a common problem when modelling with machine learning: where to switch from the machine-learned model to an established, physical one. Moreover, boundary optimisation could be applied to three-body interactions or alternative systems straightforwardly given a suitable long-range function.

Chapter 4, meanwhile, presented results of training a GP potential with alternative input and output transforms, as well as new kernels. This chapter was motivated by results from Chapter 3, namely the aforementioned instability in the hyperparameters for the $(HX)_2$ dimers when training without boundary optimisation. As this instability arose from non-stationarity in the training data, it was hoped that enhancing stationarity through new transforms or introducing a non-stationary kernel would mitigate it. However, none of these methods were successful: an alternative energy transform facilitated a meagre reduction in error on the $(HCl)_2$ PES; a composite kernel comprising the sum of a squared exponential and neural network kernel was found to slightly reduce error for larger training sets at the expense of increased training costs; and a new transform on the inputs only reduced predictive accuracy further as it diverged from the original $r \rightarrow r^{-1}$ transform. Thus, the efficiency and stability gains from boundary optimisation could not be improved upon further in this way, suggesting a combination of this technique with the current symmetric kernel and input transform is sufficient for most systems.

Therefore, in Chapter 5 a method for efficient implementation of GP potentials was explored. This method entailed parallelising the calculation of the non-additive triplet energies, which required calculation of the exponentials with shared memory and distribution of the triplet energy calculations across all processors. Though the exponential calculation was GP-specific, the method for distributing the triplets for evaluation was general to any potential and ensured the triplets were shared equitably across all processors. Moreover, parallelisation of the triplet energy evaluations is likely to be effective in enhancing the speed of the calculations as these can be calculated independently and then summed. Though evaluation of the exponentials did not parallelise particularly well, it transpired that these calculations were a small fraction of the total time taken. The dominant cost was instead the time taken to sum the exponentials to evaluate the triplet energies, which parallelised near-perfectly. Overall, a 15-fold speed-up was observed over 40 processors for the full non-additive energy calculation of a full simulation box, with a similar result obtained when finding the change in that energy when an atom moved. Over 10 processors, a roughly seven-fold speed-up was seen in both of these calculations. When calculations were spread over multiple nodes, similar speed-up was observed. In addition, as the size of the box, and hence the cut-off radius, increases, the proportion of triplet calculations will increase. This will enhance the observed speed-up further over more processors. The additive energy calculation is less computationally costly because it will require fewer exponential sums and a GP with a smaller training set. Consequently, the algorithm already addresses the largest computational barrier to accurate simulations of liquids and gases with GP potentials.

Consequently, although attempts to vary the kernel and input/output transforms employed showed no improvement to model training, the work presented here has shown two further routes for reducing the computational cost of GP potentials. The first, boundary optimisation, focuses on achieving this through a reduction in training set size, while the second approaches the problem from the perspective of exploiting the GP framework to undertake calculations efficiently. This work therefore relays methods that enhance drastically our capacity to perform first principles simulations of fluids with GP potentials.

Before such simulations can be undertaken, however, further work is required.

For the boundary optimisation, applications to three-body systems have not been attempted. These require only a three-body long-range function, which can be obtained straightforwardly by generalisation of the method in Appendix D. Otherwise, the methodology required will be identical to the two-body case. It may also be worth investigating if re-designing the training Latin hypercubes (LHCs) results in more stable training of the $(HX)_2$ potentials, though it is doubtful that this will have a significant effect.

However, if running such a simulation is desired, the focus should be on the future work described in Section 5.10. This details the requisite changes for employing GP potentials in simulations, including extensions to atomic and molecular mixtures, and the determination of forces for molecular dynamics simulations. Promisingly, all of these changes are straightforward compared with the algorithm developed so far. Moreover, said algorithm already handles the most computationally intensive aspect of the calculation, the three-body calculation. As such, the durations of the calculations upon introducing the aforementioned changes will not be drastically increased. This means it is likely that a proof-of-concept first principles simulation of liquid argon should be possible in the near future, before extension to more interesting systems and molecular dynamics simulations. Though argon is not the most abundant contaminant in CCS pipelines, it is commonly found in CCS mixtures and its simulation with a machine-learned potential would represent an important step towards applying these potentials in more representative simulations. In addition, these sections introduce a general form for evaluating all necessary three-body long-range corrections, which can be applied using a variety of long-range functions. Holistically, therefore, this work comprises methodologies to reduce significantly the computational cost of first principles simulation with GP potentials.

Given the potential importance of carbon capture and storage (CCS) in reducing emissions and mitigating the effects of climate change, the work has potentially pervasive applications. With minimal additions, all of which are described, the algorithm will be applicable to phase co-existence calculations of argon in Monte Carlo simulations using the Gibbs ensemble. With a few further modifications, such a calculation can be undertaken for atomic and molecular mixtures akin to those in a CCS pipeline. This would permit the evaluation of the vapour-liquid phase transition point in these mixtures, facilitating more efficient pipeline operation. Furthermore, once forces are employed, molecular dynamics can be used to assess non-equilibrium properties such as viscosity, which also affect pipeline operation. All of these properties would be calculated using first principles predictions, ensuring simulation results are quantitatively accurate and eliminating the need for laborious experiments to either fit potentials or evaluate the accuracy of simulation results. In addition, application of GP potentials to liquid simulations would facilitate derivation of liquid equations of state from first principles. Methods to derive such equations of state systematically from simulations [76, 77] could be used to this end for the complex mixtures found in CCS pipelines.

Appendix A

Cholesky decomposition

Consider an N x N positive definite matrix $\overline{\mathbf{A}}$. Under the Cholesky decomposition

$$\overline{\mathbf{A}} = \overline{\mathbf{R}}^{\mathrm{T}}\overline{\mathbf{R}}, \tag{A.1}$$

where $\overline{\mathbf{R}}$ is an upper-triangular matrix that is non-zero in all elements of its diagonal,

$$\overline{\mathbf{R}} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ 0 & r_{22} & \cdots & r_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{NN} \end{bmatrix}. \tag{A.2}$$

From equation (A.1), it follows that

$$\overline{\mathbf{A}}^{-1} = (\overline{\mathbf{R}}^{\mathrm{T}}\overline{\mathbf{R}})^{-1}. \tag{A.3}$$

Thus, $\overline{\mathbf{A}}^{-1}$ can be found from $\overline{\mathbf{R}}^{-1}$.

This is done by first denoting $\overline{\mathbf{R}}^{-1} = \overline{\mathbf{M}}$. Then

$$\overline{\mathbf{R}}\overline{\mathbf{M}} = \bar{\mathbf{I}}, \tag{A.4}$$

where $\bar{\mathbf{I}}$ is the identity matrix. This can be solved using forward propagation

$$\overline{\mathbf{R}}\mathbf{M}_i = \mathbf{I}_i, \tag{A.5}$$

where i denotes a column of a matrix. Once equation (A.5) has been solved for all i, $\overline{\mathbf{A}}^{-1}$ is found using

$$\overline{\mathbf{A}}^{-1} = \overline{\mathbf{M}}^{\mathrm{T}}\overline{\mathbf{M}}. \tag{A.6}$$

Appendix B

Conditioning a joint Gaussian distribution

Consider two vectors of observations $\mathbf{a}$ and $\mathbf{b}$ from a joint Gaussian distribution,

$$\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} \sim \mathcal{N}\Big(\begin{bmatrix}\boldsymbol{\mu}_{\mathbf{a}}\\ \boldsymbol{\mu}_{\mathbf{b}}\end{bmatrix}, \begin{bmatrix}\overline{\mathbf{C}} & \overline{\mathbf{D}}^{\mathrm{T}}\\ \overline{\mathbf{D}} & \overline{\mathbf{E}}\end{bmatrix}\Big). \tag{B.1}$$

Denoting as L_{a} and L_{b} the lengths of $\mathbf{a}$ and $\mathbf{b}$ respectively, $\overline{\mathbf{C}}$ is an L_{a} x L_{a} matrix, $\overline{\mathbf{D}}$ is L_{a} x L_{b} and $\overline{\mathbf{E}}$ is L_{b} x L_{b}. The conditional distribution of $\mathbf{b}$ on $\mathbf{a}$ is then [47]

$$\mathbf{b}|\mathbf{a} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{b}} + \overline{\mathbf{D}}^{\mathrm{T}}\overline{\mathbf{C}}^{-1}(\mathbf{a} - \boldsymbol{\mu}_{\mathbf{a}}), \overline{\mathbf{E}} - \overline{\mathbf{D}}^{\mathrm{T}}\overline{\mathbf{C}}^{-1}\overline{\mathbf{D}}). \tag{B.2}$$

Appendix C

Move probabilities in Gibbs Monte Carlo simulations

Monte Carlo simulations with a Gibbs ensemble entail separating the system under consideration into two distinct phases; in this case liquid and gas. For displacement moves in such simulations, the probability of acceptance is governed by

$$\Delta Y_{\mathrm{D}} = \Delta E_{\alpha} + \Delta E_{\beta}, \tag{C.1}$$

where α and β denote the different regions. Volume change moves must also account for the difference between the new and old volumes, meaning

$$\Delta Y_{\mathrm{V}} = \Delta E_{\alpha} + \Delta E_{\beta} - N_{\alpha} k_{\mathrm{B}} T \ln\Big(\frac{V_{\alpha} + \Delta V_{\alpha}}{V_{\alpha}}\Big) - N_{\beta} k_{\mathrm{B}} T \ln\Big(\frac{V_{\beta} + \Delta V_{\beta}}{V_{\beta}}\Big) + P(\Delta V_{\alpha} + \Delta V_{\beta}). \tag{C.2}$$

Finally, for a particle exchange from β to α,

$$\Delta Y_{\mathrm{V}} = \Delta E_{\alpha} + \Delta E_{\beta} + k_{\mathrm{B}} T \ln\Big(\frac{V_{\beta}(N_{\alpha} + 1)}{V_{\alpha} N_{\beta}}\Big). \tag{C.3}$$

This comes from the total number of particles across both regions being constant throughout the simulation.

Appendix D

Derivation of an empirical long-range function

HF-Na$^+$ was the only intermolecular potential examined in Chapter 3 for which a multipolar long-range function was not used. Instead a fitted, empirical long-range function was employed. Although determining an optimal long-range function was not the goal of this work, one that was accurate enough that all configurations were not rapidly transferred to the GP region was needed for proof of concept. The multipolar long-range function was unsuitable for this because of the discrepancy between its predictions and the MP2 energies.

Evidence of this is given in figure D.1, which also highlights the superiority of the predictions of the empirical long-range function. The RMSEs of each method of prediction were separated by two orders of magnitude, with the empirical function achieving an RMSE of 3.07 x 10^{-7} E_h versus 4.15 x 10^{-5} E_h for the multipolar function against the MP2 data shown in the figure.

Figure D.1 evidences that the multipolar long-range function captured the power law of the interaction energy for this system in this configuration, with the source of its error being that it was offset from the calculated energies. This offset was a product of the energies being calculated using MP2 while many of the properties used to derive the multipolar long-range function were calculated at higher levels of theory (see table 3.2). In the other systems, the small magnitude of the long-range energies meant that this disagreement was negligible and the multipolar long-range function was usable. However, in the case of HF-Na$^+$,

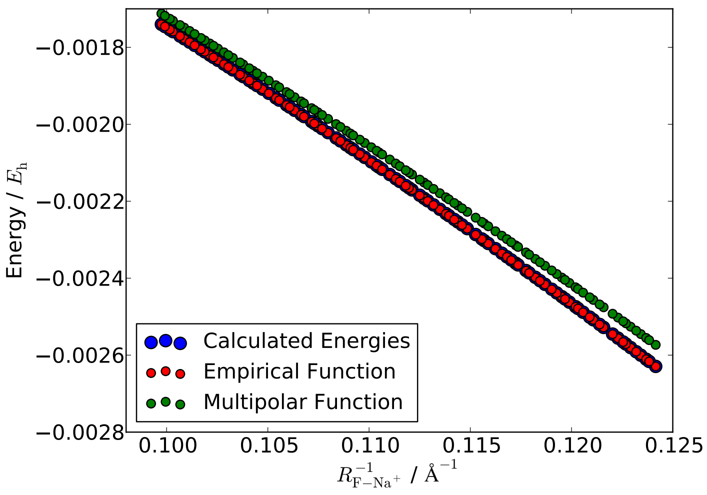


Figure D.1: Plots of the predicted energies of the long-range functions derived by the empirical method (red) and the multipole method (green) for the HF-Na$^+$ potential in a linear configuration with the F proximal to the Na$^+$ (*i.e.* with $\cos(\theta) = 1$). Also shown for comparison are MP2 energies (blue) for this configuration.

long-range energies with larger magnitudes were commonplace due to the strong repulsive and attractive interactions between the H-F dipole and the Na$^+$ cation. This exacerbated the difference between the predictions of the multipolar long-range function and the MP2 energies.

To approximate accurately the long-range data without upgrading the reference and test data to a higher level of theory, a long-range function was produced by fitting directly to these data. This was the empirical long-range function. Taking r to be the distance between the centre of the H-F bond (not centre of mass) and the Na$^+$ this function estimated the energy, E, as a sum of power laws,

$$E = \mathrm{A}r^{-2} + \mathrm{B}r^{-3}. \tag{D.1}$$

In doing so, the empirical long-range function exploits that the dominant powers of r in the HF-Na$^+$ interaction are known to be -2 and -3 but assumes that the coefficients of these terms are unknown.

As the energy varies differently with r when θ changes, a sum such as in equation (D.1) must be fitted for every configuration in a given data set for which θ is unique. For a given θ the coefficients in equation (D.1) can be found using simultaneous equations, which are set up by following algorithm 15. In all cases, $r_{\min} = 8.5$ Å and $r_{\max} = 10.5$ Å, though the power laws which resulted were accurate up to 100 Å. Furthermore, both GPs in the algorithm were trained on ~ 300 training points from a LHC design.

Algorithm 15 Coefficients for the Empirical Long-range Function

```
1: Train a GP, GP$_{\min}$, on a range of $\theta$ values at separation $r = r_{\min}$; do the same
   for another GP, GP$_{\max}$, at $r = r_{\max}$.
2: For a given $\theta$, predict $E_{\min}$ and $E_{\max}$ using GP$_{\min}$ and GP$_{\max}$ respectively.
3: Set up simultaneous equations of the form shown in equation (D.1): one with
   $E = E_{\min}$ and $r = r_{\min}$, and another with $E = E_{\max}$ and $r = r_{\max}$.
4: Solve the equations from step three for the coefficients for the current $\theta$.
```

By repeating steps 2-4 in algorithm 15, a sum of power laws was determined for every θ value in the reference and test sets. Fitting to the latter was possible as knowledge of the energies in the set over which fitting was undertaken was unnecessary. This was because the GP predictions were based on their respective training sets, and r and θ were found from the inverse interatomic separations at each configuration alone.

Deriving a long-range function from the predictions of two GPs could be problematic if a transfer learning approach were to be invoked as the data used to train these GPs would itself need to be upgraded. For the HF-Na$^+$ potential this would not be an issue because increasing the quality of the training data would increase the quality of the fit from the multipolar long-range function.

However, when an empirical function is the only option for modelling the long-range data, upgrading the data used in training GP$_{\min}$ and GP$_{\max}$ would be of considerable computational expense. This is because, currently, each comprise ~ 300 configurations. Such an issue could be circumvented by using a sequential design strategy to build minimal training sets for these GPs, which could then be upgraded instead. Furthermore, the training sets used for GP$_{\min}$ and GP$_{\max}$ are independent of that used to train a GP on the wider PES. Letting the number of training points in the training sets of GP$_{\min}$ and GP$_{\max}$ be Y and that in the training set of the other GP be Z, the cost of short-range predictions in any simulation would scale linearly with Z and the cost of any long-range predictions would scale linearly with 2Y. Given that the training of GP$_{\min}$ and GP$_{\max}$ take place at a fixed separations, it is likely that 2Y < Z if all GPs were trained under a sequential design strategy. This means that the predictions of the empirical long-range function would not be the computational bottleneck and that such a function is suitable for use in simulations. Finally, the method is sufficiently flexible that the GP predictions can easily be replaced by those of another statistical method.

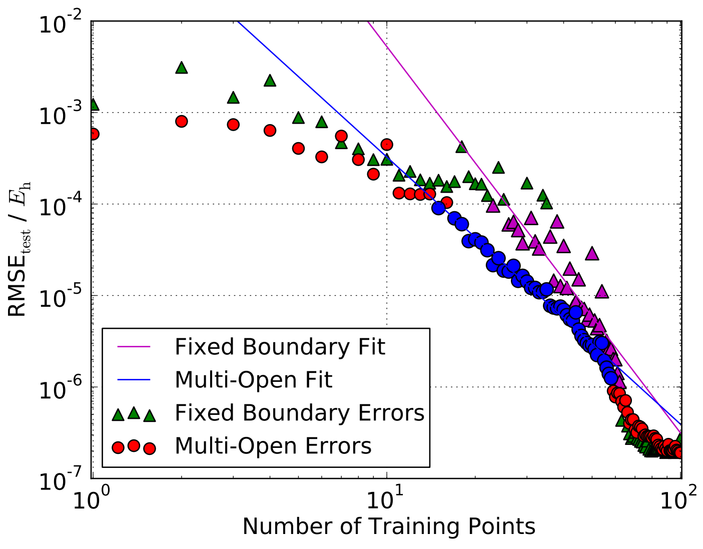
Appendix E

Boundary optimisation supplementary material

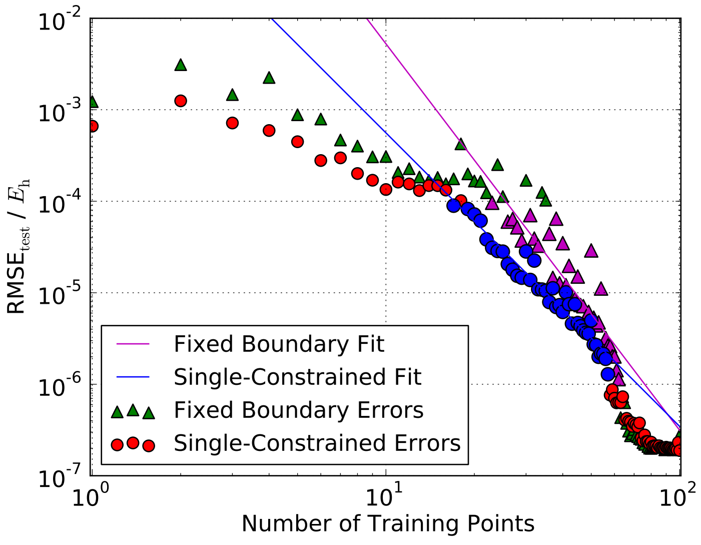
E.1 Fit Plots

This section includes the plots of $\mathrm{RMSE}_{\mathrm{test}}$ against N_{t} and the fits made to them for all training strategies for all systems. On each plot, the fit from fixed boundary training is shown for comparison.

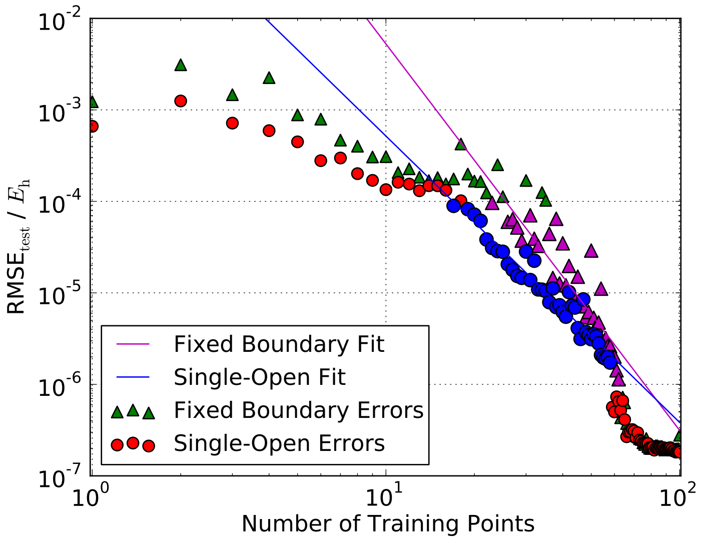
E.1.1 HF-Ne



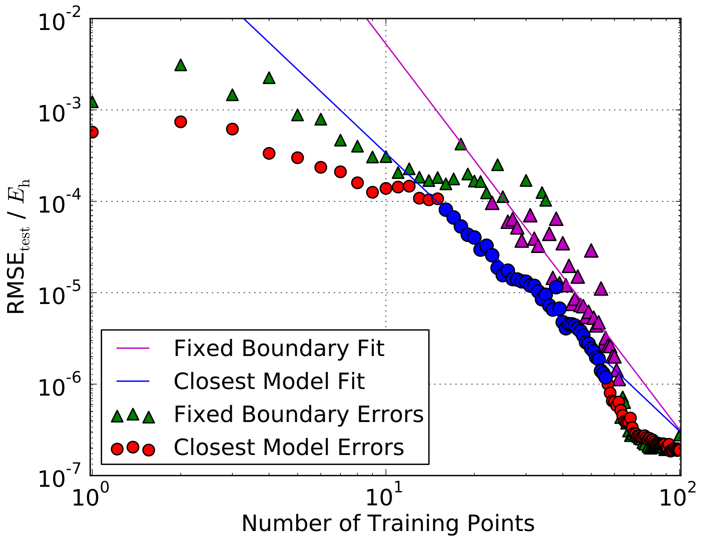
Multi-open training.



Single-constrained training.

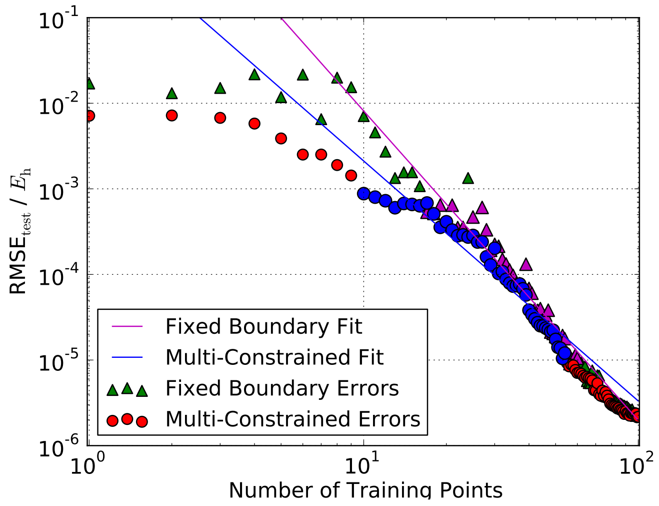


Single-open training.

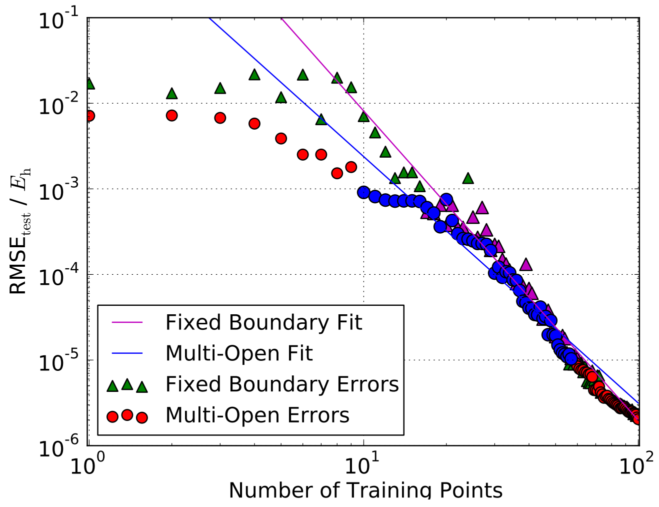


Closest model training.

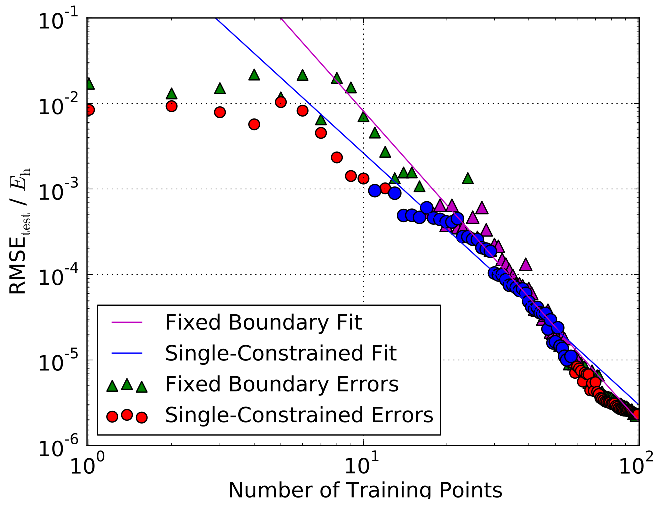
E.1.2 HF-Na$^+$



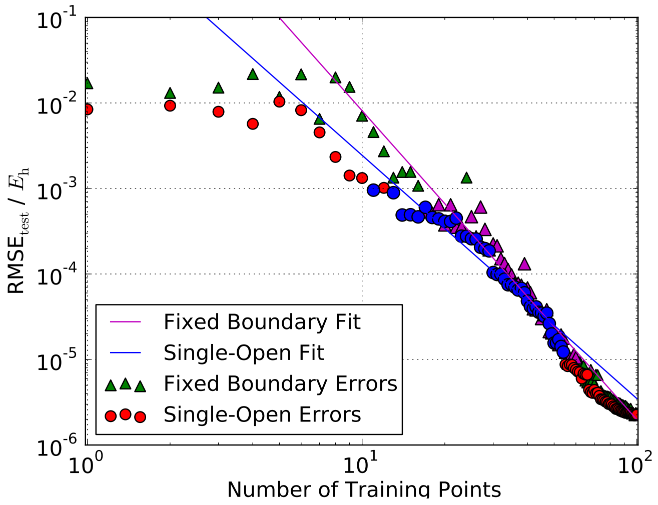
Multi-constrained training.



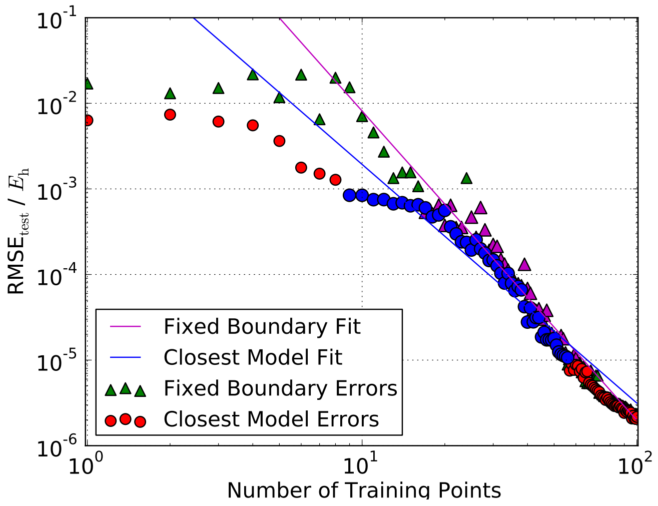
Multi-open training.



Single-constrained training.

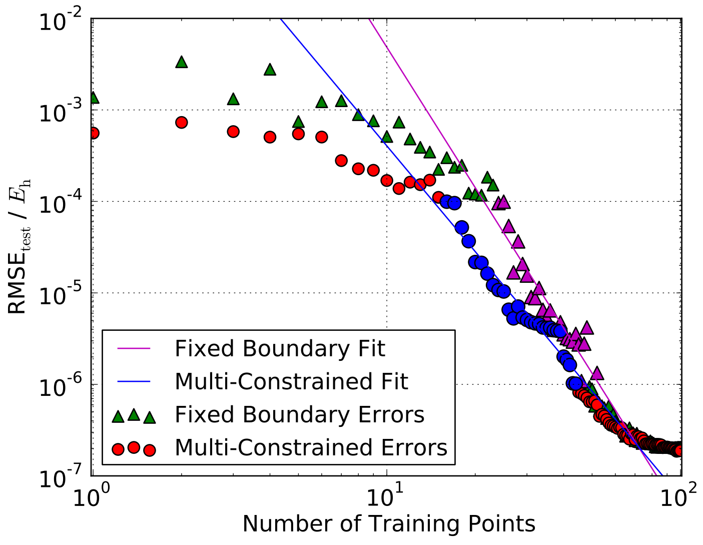


Single-open training.

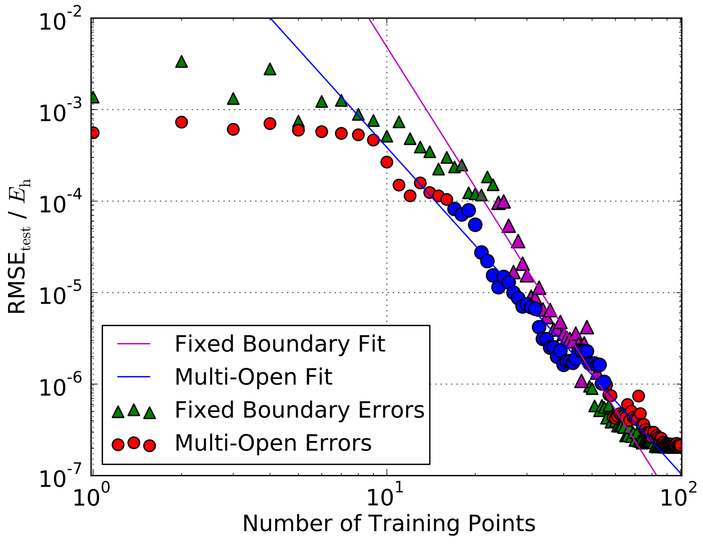


Closest model training.

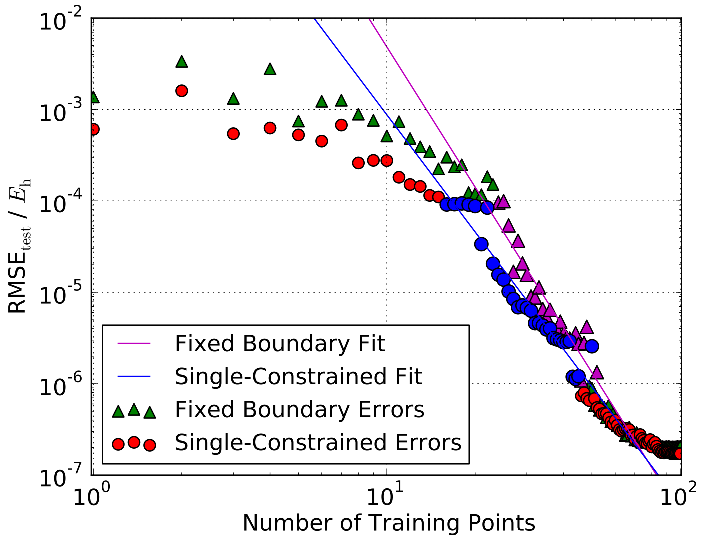
E.1.3 CO-Ne



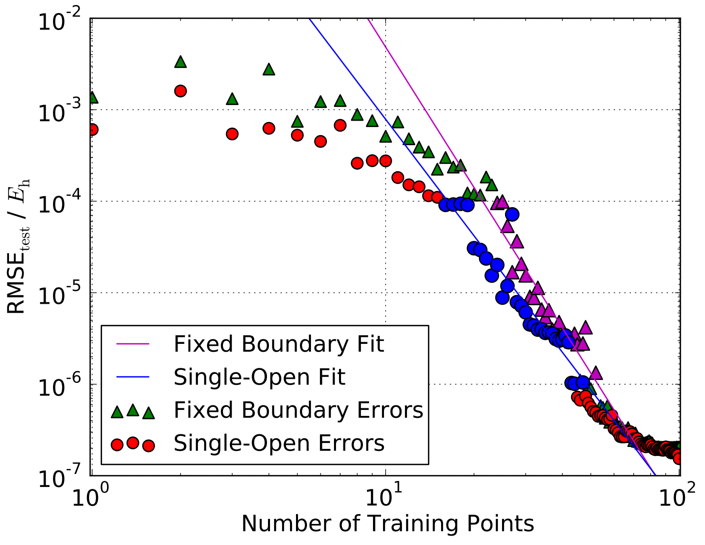
Multi-constrained training.



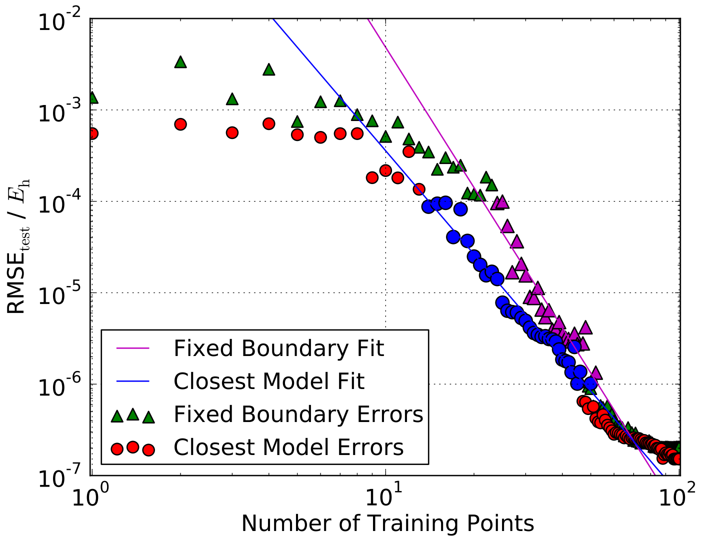
Multi-open training.



Single-constrained training.

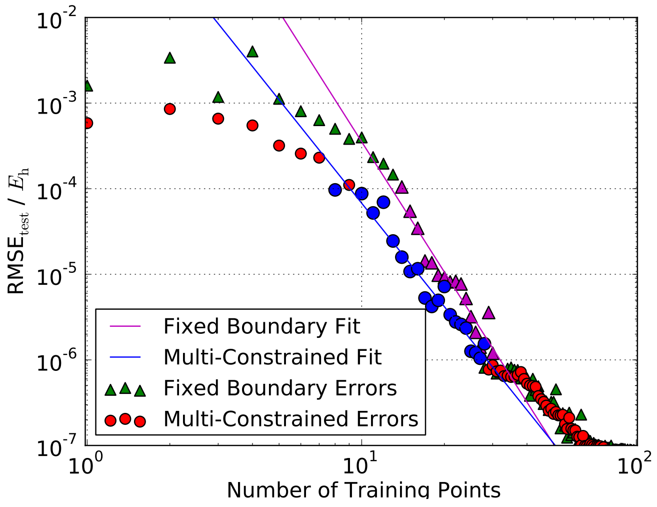


Single-open training.

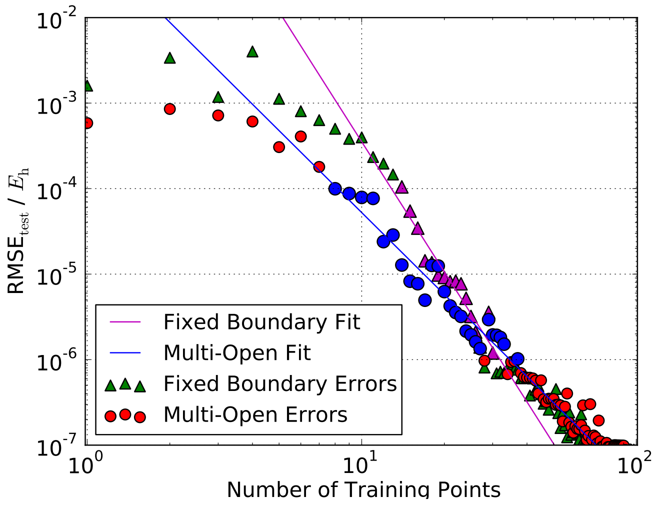


Closest model training.

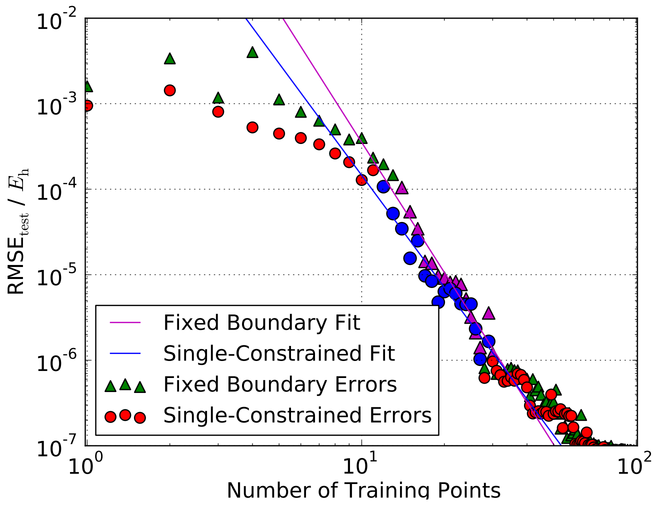
E.1.4 CO_2-Ne



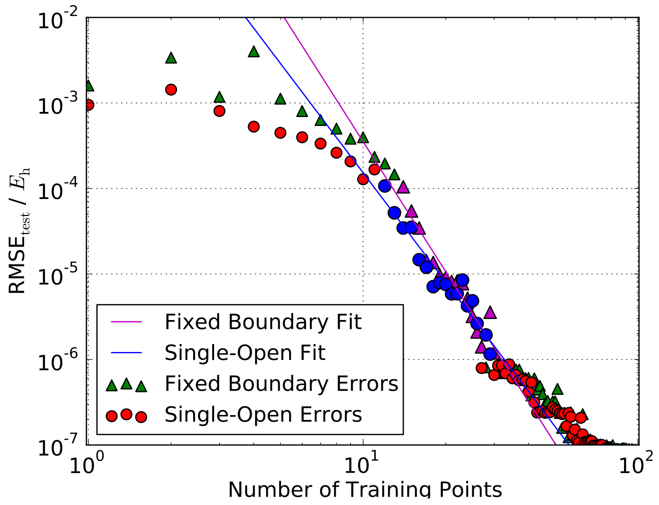
Multi-constrained training.



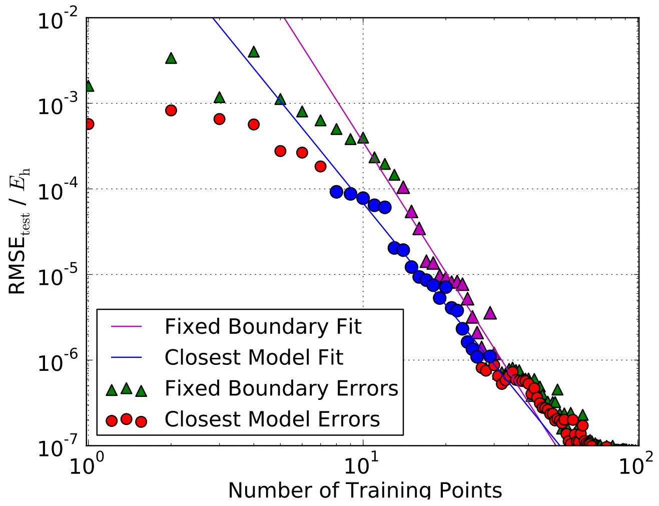
Multi-open training.



Single-constrained training.

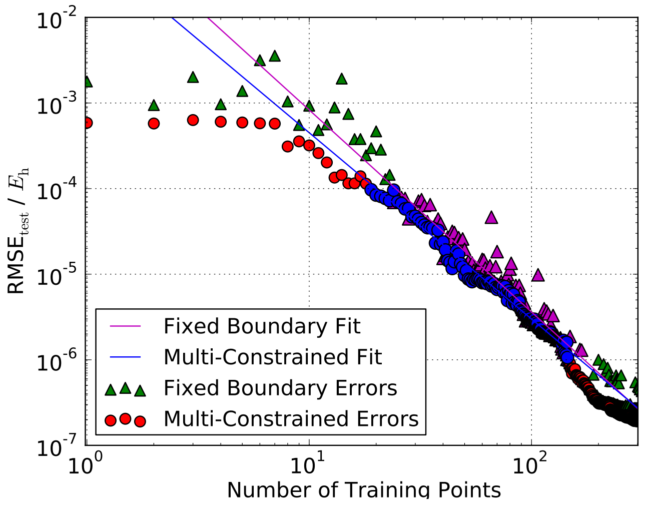


Single-open training.

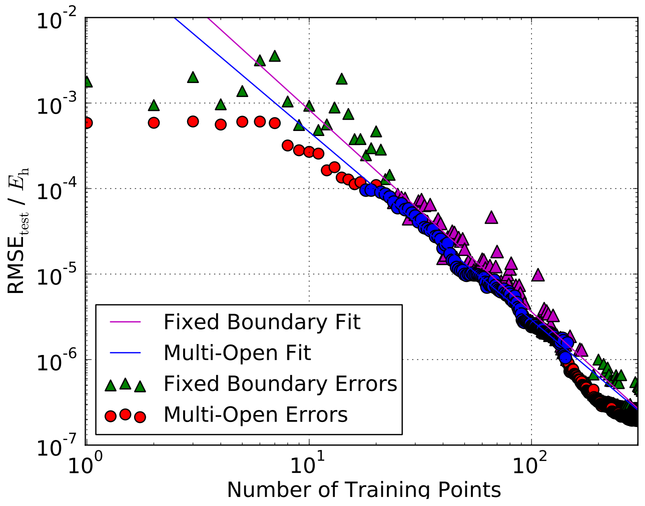


Closest model training.

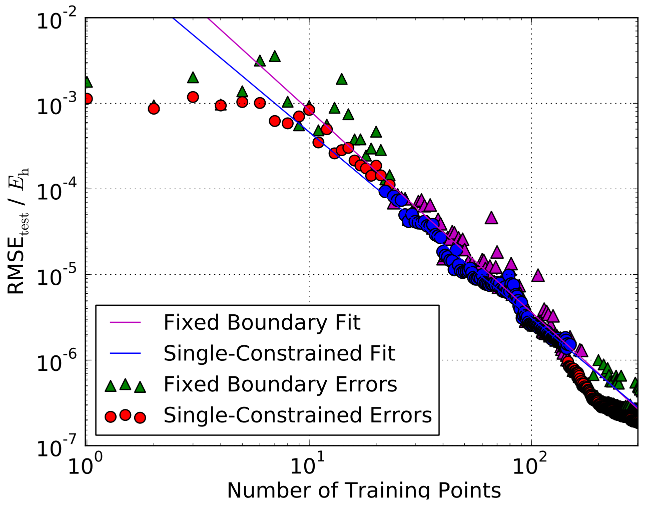
E.1.5 $(CO_2)_2$



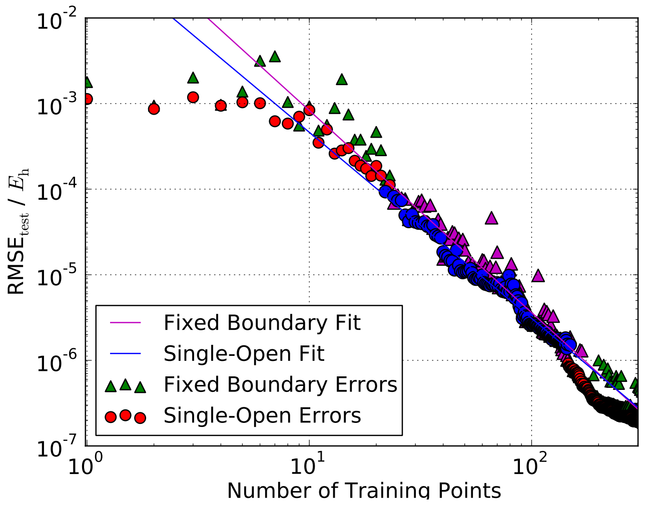
Multi-constrained training.



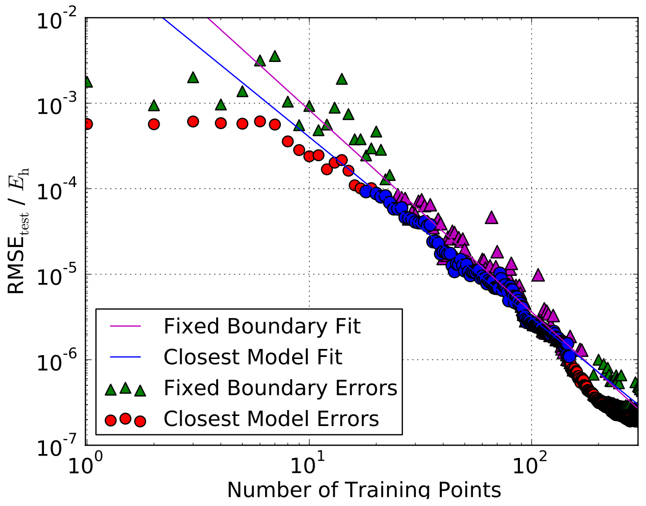
Multi-open training.



Single-constrained training.



Single-open training.



Closest model training.

E.2 Power Law Tables

This section comprises tables showing the equations of the fitted lines as power laws in the number of training points, N_t, for all training strategies for the HF-Ne, HF-Na$^+$, $(CO_2)_2$ and CO_2-Ne potentials.

HF-Ne power laws.

Training Strategy	Best fit to $\text{RMSE}_{\text{test}}$	R^2
Single-Constrained Placement	$0.383N_{\text{t}}^{-2.989}$	0.977
Multi-Constrained Placement	$0.239N_{\text{t}}^{-2.8947}$	0.920
Single-Open Placement	$0.321N_{\text{t}}^{-2.933}$	0.971
Multi-Open Placement	$0.210N_{\text{t}}^{-2.853}$	0.954
Closest Model	$0.267N_{\text{t}}^{-2.971}$	0.986
Fixed Boundary	$767.2N_{\text{t}}^{-4.791}$	0.901

HF-Na^+ power laws.

Training Strategy	Best fit to $\text{RMSE}_{\text{test}}$	R^2
Single-Constrained Placement	$2.246N_{\text{t}}^{-2.933}$	0.941
Multi-Constrained Placement	$1.343N_{\text{t}}^{-2.803}$	0.941
Single-Open Placement	$1.711N_{\text{t}}^{-2.845}$	0.944
Multi-Open Placement	$1.799N_{\text{t}}^{-2.878}$	0.931
Closest Model	$1.188N_{\text{t}}^{-2.789}$	0.920
Fixed Boundary	$33.29N_{\text{t}}^{-3.613}$	0.930

$(\text{CO}_2)_2$ power laws.

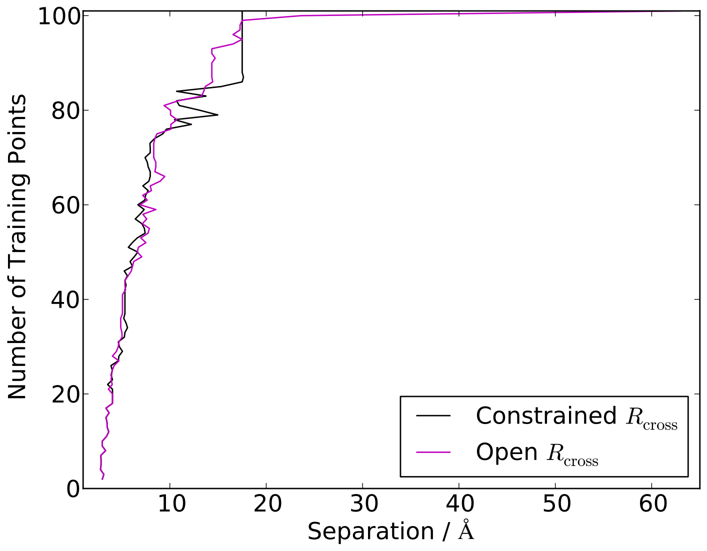
Training Strategy	Best fit to $\text{RMSE}_{\text{test}}$	R^2
Single-Constrained Placement	$0.069N_{\text{t}}^{-2.174}$	0.977
Multi-Constrained Placement	$0.064N_{\text{t}}^{-2.162}$	0.983
Single-Open Placement	$0.069N_{\text{t}}^{-2.174}$	0.977
Multi-Open Placement	$0.073N_{\text{t}}^{-2.201}$	0.988
Closest Model	$0.052N_{\text{t}}^{-2.115}$	0.984
Fixed Boundary	$0.191N_{\text{t}}^{-2.361}$	0.933

CO_2-Ne power laws.

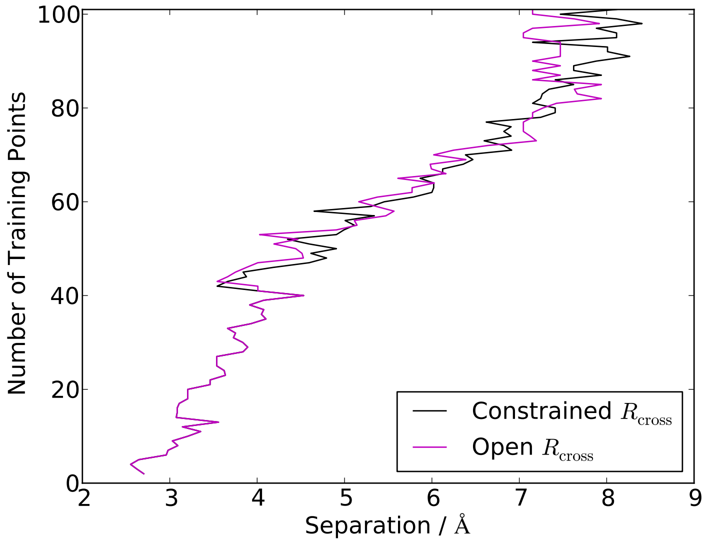
Training Strategy	Best fit to $\mathrm{RMSE}_{\mathrm{test}}$	R^2
Single-Constrained Placement	$3.237N_{\mathrm{t}}^{-4.348}$	0.910
Multi-Constrained Placement	$0.698N_{\mathrm{t}}^{-4.014}$	0.952
Single-Open Placement	$2.706N_{\mathrm{t}}^{-4.248}$	0.929
Multi-Open Placement	$0.080N_{\mathrm{t}}^{-3.184}$	0.925
Closest Model	$0.615N_{\mathrm{t}}^{-3.955}$	0.964
Fixed Boundary	$39.81N_{\mathrm{t}}^{-5.053}$	0.914

E.3 Cross-over Plots

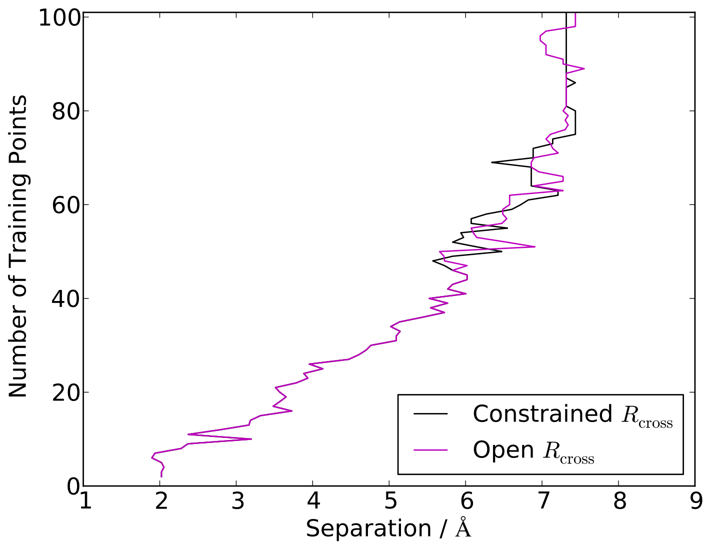
This section includes the plots of the R_{cross} values achieved by single-open and single-constrained training for all systems.



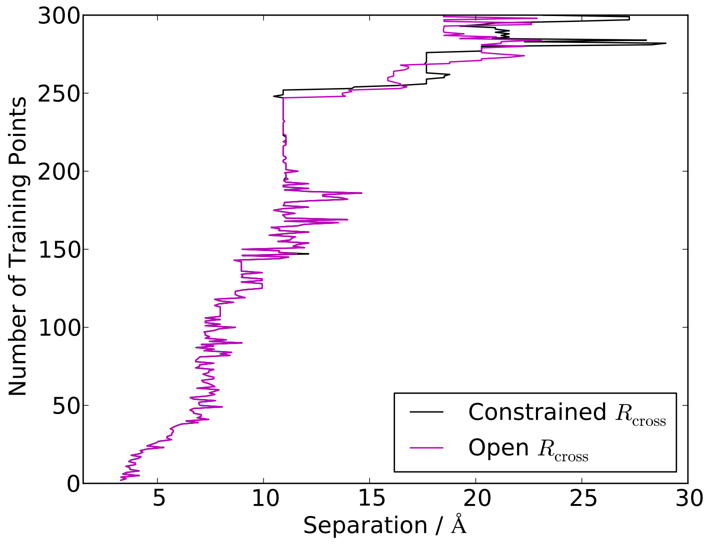
CO-Ne.



HF-Ne.



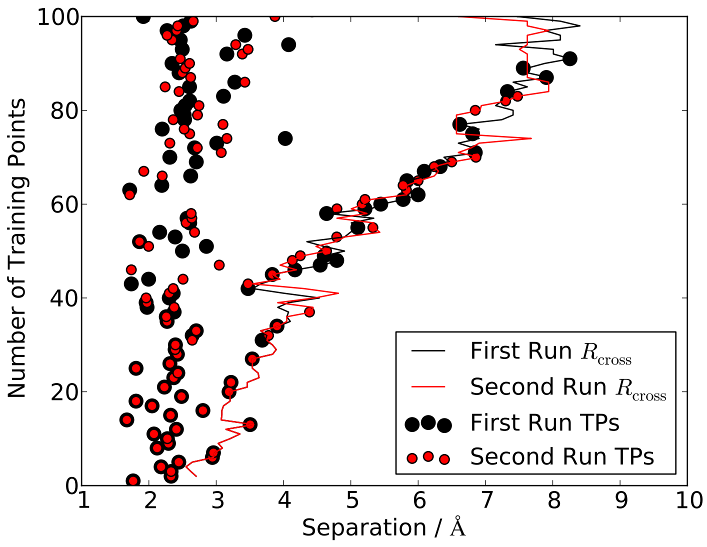
HF-Na$^+$.



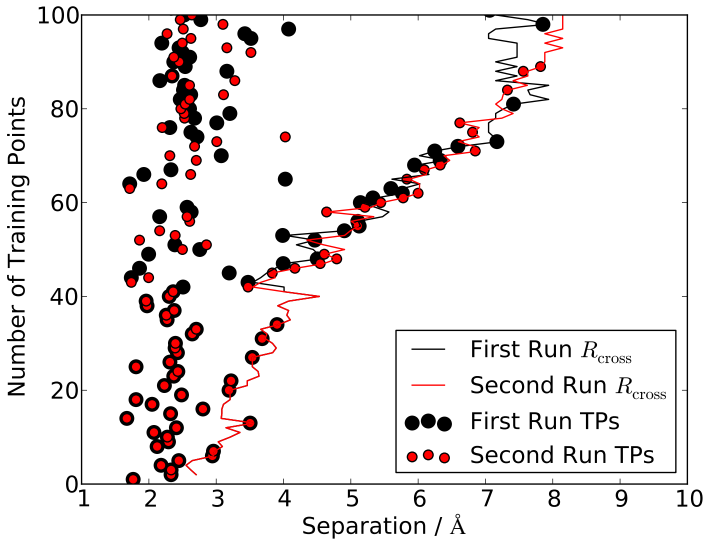
$(CO_2)_2$.

E.4 HF-Ne Reproducibility Plots

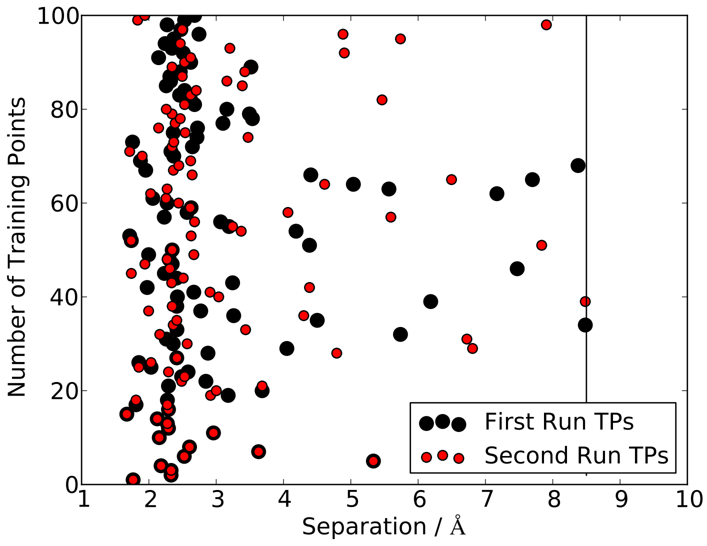
This section contains plots of the training point placement and cross-over values for two separate models from each of the single-constrained, single-open and fixed boundary training strategies on the HF-Ne system.



Single-constrained training.



Single-open training.



Fixed boundary training.

Appendix F

Parallel programming

When programming in any language, a script is produced for interpretation by a computer. For the duration of this discussion, the 'interpreter' will be referred to as a process. Often when programming, a single process is used for a single script, as outlined in figure F.1.

However, there are instances where having multiple processes interpret a single script will facilitate an increase in the speed with which that script is executed. Consider a program that requires a summation over a vector $\mathbf{V}$ of length L_V to return a value V_{sum},

$$V_{\text{sum}} = \sum_{i=1}^{L_V} \mathbf{V}_i. \tag{F.1}$$

When L_V is large calculating V_{sum} using two processes, each of which undertakes half of the sum in equation (F.1), will reduce the time taken considerably, by up to a factor of two.

In such a case, the two (or more) processes that evaluate part of the sum do so independently, only sharing their total contributions to V_{sum} in order to obtain its final value. That is, in this example, each process cannot 'see' what any other process is doing. This technique is referred to here as parallel programming and is shown schematically in figure F.2.

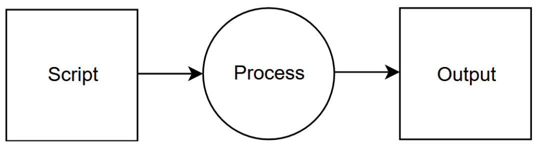


Figure F.1: A schematic showing execution of a script with one process.

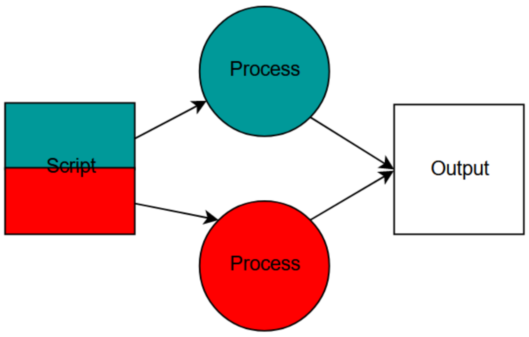


Figure F.2: A schematic showing execution of a script via parallel programming with two processes.

It is possible, though, that large amounts of information will need to be shared between processes when programming in parallel. As such, the gain in speed in executing the code could be undermined by the time taken to share this information. The solution to this is shared memory. (Though shared memory is a form of parallel programming, the two methods will be referred to as though separate for ease of discussion.) Like parallel programming, using shared memory entails having multiple processes execute a script simultaneously. However, the latter allows each process to 'see' the work done by all other processes as it is completed. Thus there is no need for any explicit sharing of information between processes, as illustrated in figure F.3.

As a result of the difference in the access each process has to the information produced by all others when using parallel programming or shared memory, both techniques are impacted differently when used over multiple nodes. A node is a server that hosts several processes, and only processes on the same node can share memory. Thus, any shared memory method must be undertaken separately on each node. Meanwhile, under parallel programming, processes on different nodes can still send information to each other. The time taken to do so may be longer than on a single node, however. Effectively this means that parallel programming strategies are affected barely at all by implementation across various nodes, unlike those that use shared memory.

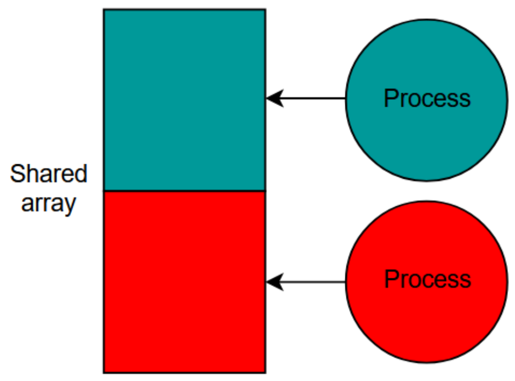


Figure F.3: A schematic showing an array being filled by two processes that share memory.

Bibliography

[1] T. A. Halgren and W. Damm, "Polarizable force fields," *Curr. Opin. Struct. Biol.*, vol. 11, no. 2, pp. 236–242, 2001.

[2] A. Warshel, M. Kato, and A. V. Pisliakov, "Polarizable force fields: history, test cases, and prospects," *J. Chem. Theory Comput.*, vol. 3, no. 6, pp. 2034–2045, 2007.

[3] P. Ren and J. W. Ponder, "Polarizable atomic multipole water model for molecular mechanics simulation," *J. Phys. Chem. B*, vol. 107, no. 24, pp. 5933–5947, 2003.

[4] A. Grossfield, P. Ren, and J. W. Ponder, "Ion solvation thermodynamics from simulation with a polarizable force field," *J. Am. Chem. Soc.*, vol. 125, no. 50, pp. 15671–15682, 2003.

[5] P. Ren, C. Wu, and J. W. Ponder, "Polarizable atomic multipole-based molecular mechanics for organic molecules," *J. Chem. Theory Comput.*, vol. 7, no. 10, pp. 3143–3161, 2011.

[6] C. Liu, J.-P. Piquemal, and P. Ren, "AMOEBA+ classical potential for modeling molecular interactions," *J. Chem. Theory Comput.*, vol. 15, no. 7, pp. 4122–4139, 2019.

[7] C. Liu, J.-P. Piquemal, and P. Ren, "Implementation of geometry-dependent charge flux into the polarizable AMOEBA+ potential," *J. Phys. Chem. Lett.*, vol. 11, no. 2, pp. 419–426, 2019.

[8] C. Zhang, C. Lu, Z. Jing, C. Wu, J.-P. Piquemal, J. W. Ponder, and P. Ren, "AMOEBA polarizable atomic multipole force field for nucleic acids," *J. Chem. Theory Comput.*, vol. 14, no. 4, pp. 2084–2108, 2018.

[9] Z. Jing, R. Qi, C. Liu, and P. Ren, "Study of interactions between metal ions and protein model compounds by energy decomposition analyses and the AMOEBA force field," *J. Chem. Phys.*, vol. 147, no. 16, p. 161733, 2017.

[10] Z. Jing, C. Liu, R. Qi, and P. Ren, "Many-body effect determines the selectivity for Ca^{2+} and Mg^{2+} in proteins," *PNAS*, vol. 115, no. 32, pp. E7495–E7501, 2018.

[11] P. Xu, E. B. Guidez, C. Bertoni, and M. S. Gordon, "Perspective: *Ab initio* force field methods derived from quantum mechanics," *J. Chem. Phys.*, vol. 148, no. 9, p. 090901, 2018.

[12] I. Adamovic, M. A. Freitag, and M. S. Gordon, "Density functional theory based effective fragment potential method," *J. Chem. Phys.*, vol. 118, no. 15, pp. 6725–6732, 2003.

[13] J.-P. Piquemal, H. Chevreau, and N. Gresh, "Toward a Separate Reproduction of the Contributions to the Hartree- Fock and DFT Intermolecular Interaction Energies by Polarizable Molecular Mechanics with the SIBFA Potential," *J. Chem. Theory Comput.*, vol. 3, no. 3, pp. 824–837, 2007.

[14] J. Behler, "Perspective: Machine learning potentials for atomistic simulations," *J. Chem. Phys*, vol. 145, no. 17, p. 170901, 2016.

[15] T. Mueller, A. Hernandez, and C. Wang, "Machine learning for interatomic potential models," *J. Chem. Phys.*, vol. 152, no. 5, p. 050902, 2020.

[16] B. Selvaratnam and R. T. Koodali, "Machine Learning in Experimental Materials Chemistry," *Catal. Today*, vol. 371, pp. 77–84, 2021.

[17] J. Noh, G. H. Gu, S. Kim, and Y. Jung, "Machine-enabled inverse design of inorganic solid materials: promises and challenges," *Chem. Sci.*, vol. 11, no. 19, pp. 4871–4881, 2020.

[18] P. O. Dral, "Quantum chemistry in the age of machine learning," *J. Phys. Chem. Lett.*, vol. 11, no. 6, pp. 2336–2347, 2020.

[19] J. Behler and M. Parrinello, "Generalized neural-network representation of high-dimensional potential-energy surfaces," *Phys. Rev. Lett.*, vol. 98, no. 14, p. 146401, 2007.

[20] N. Artrith, T. Morawietz, and J. Behler, "High-dimensional neural-network potentials for multicomponent systems: Applications to zinc oxide," *Phys. Rev. B*, vol. 83, no. 15, p. 153101, 2011.

[21] N. Artrith, B. Hiller, and J. Behler, "Neural network potentials for metals and oxides - First applications to copper clusters at zinc oxide," *Phys. Status Solidi B*, vol. 250, no. 6, pp. 1191–1203, 2013.

[22] J. Behler, "Representing potential energy surfaces by high-dimensional neural network potentials," *J. Phys. Condens. Matt.*, vol. 26, no. 18, p. 183001, 2014.

[23] Q. Lin, Y. Zhang, B. Zhao, and B. Jiang, "Automatically growing global reactive neural network potential energy surfaces: a trajectory free active learning strategy," *J. Chem. Phys.*, vol. 152, no. 15, p. 154104, 2020.

[24] A. V. Shapeev, "Moment tensor potentials: A class of systematically improvable interatomic potentials," *MMS*, vol. 14, no. 3, pp. 1153–1173, 2016.

[25] E. V. Podryabinkin and A. V. Shapeev, "Active learning of linearly parametrized interatomic potentials," *Comput. Mater. Sci.*, vol. 140, pp. 171–180, 2017.

[26] I. I. Novoselov, A. V. Yanilkin, A. V. Shapeev, and E. V. Podryabinkin, "Moment tensor potentials as a promising tool to study diffusion processes," *Comput. Mater. Sci.*, vol. 164, pp. 46–56, 2019.

[27] E. Uteva, R. J. Wheatley, R. D. Wilkinson, and R. S. Graham, "Interpolation of intermolecular potentials using Gaussian processes," *J. Chem. Phys.*, vol. 147, p. 161706, 2017.

[28] A. J. Cresswell, R. J. Wheatley, R. D. Wilkinson, and R. S. Graham, "Molecular simulation of the thermophysical properties and phase behaviour of impure CO_2 relevant to CCS," *Faraday Discuss.*, vol. 192, pp. 415–436, 2016.

[29] E. Uteva, R. J. Wheatley, R. D. Wilkinson, and R. S. Graham, "Active learning in Gaussian process interpolation of potential energy surfaces," *J. Chem. Phys.*, vol. 149, p. 174114, 2018.

[30] R. S. Graham and R. J. Wheatley, "Machine learning for non-additive intermolecular potentials: from quantum chemistry to first-principles predictions," *ChemComm*, 2022.

[31] C. M. Handley, G. I. Hawe, D. B. Kell, and P. L. A. Popelier, "Optimal construction of a fast and accurate polarisable water potential based on multipole moments trained by machine learning," *Phys. Chem. Chem. Phys.*, vol. 11, no. 30, pp. 6365–6376, 2009.

[32] M. J. L. Mills and P. L. A. Popelier, "Intramolecular polarisable multipolar electrostatics from the machine learning method Kriging," *Comput. Theor. Chem.*, vol. 975, no. 1-3, pp. 42–51, 2011.

[33] M. J. L. Mills and P. L. A. Popelier, "Polarisable multipolar electrostatics from the machine learning method Kriging: an application to alanine," *Theor. Chem. Acc.*, vol. 131, no. 3, p. 1137, 2012.

[34] S. M. Kandathil, T. L. Fletcher, Y. Yuan, J. Knowles, and P. L. A. Popelier, "Accuracy and tractability of a Kriging model of intramolecular polarizable multipolar electrostatics and its application to histidine," *J. Comput. Chem.*, vol. 34, no. 21, pp. 1850–1861, 2013.

[35] J. Dai and R. V. Krems, "Interpolation and extrapolation of global potential energy surfaces for polyatomic systems by Gaussian processes with composite kernels," *J. Chem. Theory Comput.*, vol. 16, no. 3, pp. 1386–1395, 2020.

[36] H. Sugisawa, T. Ida, and R. V. Krems, "Gaussian process model of 51-dimensional potential energy surface for protonated imidazole dimer," *J. Chem. Phys.*, vol. 153, no. 11, p. 114101, 2020.

[37] G. Schmitz, E. L. Klinting, and O. Christiansen, "A Gaussian process regression adaptive density guided approach for potential energy surface construction," *J. Chem. Phys.*, vol. 153, no. 6, p. 064105, 2020.

[38] M. J. Burn and P. L. A. Popelier, "Creating Gaussian process regression models for molecular simulations using adaptive sampling," *J. Chem. Phys.*, vol. 153, no. 5, p. 054111, 2020.

[39] M. A. Boussaidi, O. Ren, D. Voytsekhovsky, and S. Manzhos, "Random Sampling High Dimensional Model Representation Gaussian Process Regression (RS-HDMR-GPR) for multivariate function representation: application to molecular potential energy surfaces," *J. Phys. Chem. A*, vol. 124, no. 37, pp. 7598–7607, 2020.

[40] G. Sivaraman, A. N. Krishnamoorthy, M. Baur, C. Holm, M. Stan, G. Csányi, C. Benmore, and Á. Vázquez-Mayagoitia, "Machine-learned interatomic potentials by active learning: amorphous and liquid hafnium dioxide," *Npj Comput. Mater.*, vol. 6, no. 1, pp. 1–8, 2020.

[41] M. A. Caro, G. Csanyi, T. Laurila, and V. L. Deringer, "Machine learning driven simulated deposition of carbon films: From low-density to diamond-like amorphous carbon," *Phys. Rev. B.*, vol. 102, no. 17, p. 174201, 2020.

[42] Y. B. Liu, J. Y. Yang, G. M. Xin, L. H. Liu, G. Csányi, and B. Y. Cao, "Machine learning interatomic potential developed for molecular simulations on thermal properties of β-Ga_2O_3," *J. Chem. Phys.*, vol. 153, no. 14, p. 144501, 2020.

[43] A. Glielmo, P. Sollich, and A. De Vita, "Accurate interatomic force fields via machine learning with covariant kernels," *Phys. Rev. B*, vol. 95, no. 21, p. 214302, 2017.

[44] A. Glielmo, C. Zeni, and A. De Vita, “Efficient nonparametric n-body force fields from machine learning,” *Phys. Rev. B*, vol. 97, no. 18, p. 184307, 2018.

[45] C. Zeni, K. Rossi, A. Glielmo, Á. Fekete, N. Gaston, F. Baletto, and A. De Vita, “Building machine learning force fields for nanoclusters,” *J. Chem. Phys.*, vol. 148, no. 24, p. 241739, 2018.

[46] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, “Neural network models of potential energy surfaces,” *J. Chem. Phys.*, vol. 103, no. 10, pp. 4129–4137, 1995.

[47] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[48] N. Cressie, “The origins of Kriging,” *Math. Geol.*, vol. 22, no. 3, pp. 239–252, 1990.

[49] A. Kamath, R. A. Vargas-Hernández, R. V. Krems, T. Carrington Jr, and S. Manzhos, “Neural networks vs Gaussian process regression for representing potential energy surfaces: A comparative study of fit quality and vibrational spectrum accuracy,” *J. Chem. Phys.*, vol. 148, no. 24, p. 241702, 2018.

[50] Y. Zuo, C. Chen, X. Li, Z. Deng, Y. Chen, J. Behler, G. Csányi, A. V. Shapeev, A. P. Thompson, M. A. Wood, *et al.*, “Performance and Cost Assessment of Machine Learning Interatomic Potentials,” *J. Phys. Chem. A*, vol. 124, no. 4, pp. 731–745, 2020.

[51] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pp. 242–264, IGI global, 2010.

[52] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2009.

[53] P. Maxwell, N. di Pasquale, S. Cardamone, and P. L. A. Popelier, “The prediction of topologically partitioned intra-atomic and inter-atomic ener-

gies by the machine learning method Kriging," *Theor. Chem. Acc.*, vol. 135, no. 8, p. 195, 2016.

[54] P. I. Maxwell and P. L. A. Popelier, "Accurate prediction of the energetics of weakly bound complexes using the machine learning method Kriging," *Struct. Chem.*, vol. 28, no. 5, pp. 1513–1523, 2017.

[55] Z. E. Hughes, E. Ren, J. C. R. Thacker, B. C. B. Symons, A. F. Silva, and P. L. A. Popelier, "A FFLUX water model: Flexible, Polarizable and with a Multipolar Description of Electrostatics," *J. Comput. Chem.*, vol. 41, no. 7, pp. 619–628, 2020.

[56] T. L. Fletcher and P. L. A. Popelier, "FFLUX: Transferability of polarizable machine-learned electrostatics in peptide chains," *J. Comput. Chem.*, vol. 38, no. 13, pp. 1005–1014, 2017.

[57] J. C. R. Thacker, A. L. Wilson, Z. E. Hughes, M. J. Burn, P. I. Maxwell, and P. L. A. Popelier, "Towards the simulation of biomolecules: optimisation of peptide-capped glycine using FFLUX," *Mol. Simul.*, vol. 44, no. 11, pp. 881–890, 2018.

[58] A. Konovalov, B. C. B. Symons, and P. L. A. Popelier, "On the many-body nature of intramolecular forces in FFLUX and its implications," *J. Comput. Chem.*, vol. 42, no. 2, pp. 107–116, 2020.

[59] K. R. Bartók, A P and G. Csányi, "On representing chemical environments," *Phys. Rev. B*, vol. 87, no. 18, p. 184115, 2013.

[60] F. C. Mocanu, K. Konstantinou, T. H. Lee, N. Bernstein, V. L. Deringer, G. Csányi, and S. R. Elliott, "Modeling the phase-change memory material, $Ge_2Sb_2Te_5$, with a machine-learned interatomic potential," *J. Phys. Chem. B*, vol. 122, no. 38, pp. 8998–9006, 2018.

[61] D. Dragoni, T. D. Daff, G. Csányi, and N. Marzari, "Achieving DFT accuracy with a machine-learning interatomic potential: Thermomechanics and defects in BCC ferromagnetic iron," *Phys. Rev. Mater.*, vol. 2, no. 1, p. 013808, 2018.

[62] A. P. Bartók, J. Kermode, N. Bernstein, and G. Csányi, "Machine learning a general-purpose interatomic potential for silicon," *Phys. Rev. X*, vol. 8, no. 4, p. 041048, 2018.

[63] V. L. Deringer, M. A. Caro, and G. Csányi, "A general-purpose machine-learning force field for bulk and nanostructured phosphorus," *Nat. Commun.*, vol. 11, no. 1, pp. 1–11, 2020.

[64] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[65] M. Stein, "Large sample properties of simulations using Latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[66] W.-L. Loh, "On Latin hypercube sampling," *Ann. Stat.*, vol. 24, no. 5, pp. 2058–2080, 1996.

[67] H. Robbins, "Some aspects of the sequential design of experiments," *Bull. Am. Math. Soc.*, vol. 58, no. 5, pp. 527–535, 1952.

[68] H. Chernoff, "Sequential design of experiments," *Ann. Math. Stat.*, vol. 30, no. 3, pp. 755–770, 1959.

[69] D. Bowler and T. Miyazaki, "$\mathcal{O}(N)$ methods in electronic structure calculations," *Rep. Prog. Phys.*, vol. 75, no. 3, p. 036503, 2012.

[70] S. Mohr, M. Eixarch, M. Amsler, M. J. Mantsinen, and L. Genovese, "Linear scaling DFT calculations for large tungsten systems using an optimized local basis," *Nucl. Mater. Energy*, vol. 15, pp. 64–70, 2018.

[71] "Accelerating Breakthrough Innovation in Carbon Capture, Utilization and Storage." https://www.energy.gov/fecm/downloads/accelerating-breakthroughinnovation- carbon-capture-utilization-and-storage. Accessed: 27/06/2022.

[72] "Re-affirming our commitment to deploying CCUS in the UK subject to cost reduction." https://www.gov.uk/guidance/uk-carbon-capture-and-storage-government-funding-and-support. Accessed: 27/06/2022.

[73] J. H. Dymond, K. N. Marsh, R. C. Wilhoit, M. D. Frenkel, *et al.*, *Virial Coefficients of Mixtures.* Springer-Verlag, 2021.

[74] P. Czachorowski, M. Przybytek, M. Lesiuk, M. Puchalski, and B. Jeziorski, "Second virial coefficients for He_4 and He_3 from an accurate relativistic interaction potential," *Phy. Rev. A*, vol. 102, no. 4, p. 042810, 2020.

[75] R. Hellmann, "Reference values for the second virial coefficient and three dilute gas transport properties of ethane from a state-of-the-art intermolecular potential energy surface," *J. Chem. Eng. Data*, vol. 63, no. 2, pp. 470–481, 2018.

[76] H. Do, J. D. Hirst, and R. J. Wheatley, "Rapid calculation of partition functions and free energies of fluids," *J. Chem. Phys.*, vol. 135, no. 17, p. 174105, 2011.

[77] M. Veit, S. K. Jain, S. Bonakala, I. Rudra, D. Hohl, and G. Csányi, "Equation of state of fluid methane from first principles with machine learning potentials," *J. Chem. Theory Comput.*, vol. 15, no. 4, pp. 2574–2586, 2019.

[78] L. Yu, F. Nina-Paravecino, D. R. Kaeli, and Q. Fang, "Scalable and massively parallel Monte Carlo photon transport simulations for heterogeneous computing platforms," *J. Biomed. Opt.*, vol. 23, no. 1, p. 010504, 2018.

[79] H. Fan and S. Li, "Parallel peridynamics–SPH simulation of explosion induced soil fragmentation by using OpenMP," *Comput. Part. Mech.*, vol. 4, no. 2, pp. 199–211, 2017.

[80] J. Latt, C. Coreixas, and J. Beny, "Cross-platform programming model for many-core lattice Boltzmann simulations," *PLoS One*, vol. 16, no. 4, p. e0250306, 2021.

[81] R. Li, H. Hu, H. Li, Y. Wu, and J. Yang, "MapReduce parallel programming model: a state-of-the-art survey," *Int. J. Parallel Program.*, vol. 44, no. 4, pp. 832–866, 2016.

[82] Y. Zhang and X. Xu, "Fe-based superconducting transition temperature modeling through Gaussian process regression," *J. Low Temp. Phys.*, vol. 202, no. 1, pp. 205–218, 2021.

[83] Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart, "Solving and learning nonlinear PDEs with Gaussian processes," *J. Comput. Phys.*, vol. 447, p. 110668, 2021.

[84] A. Damianou and N. D. Lawrence, "Deep Gaussian processes," in *Artificial intelligence and statistics*, pp. 207–215, PMLR, 2013.

[85] R. M. Neal, "Priors for infinite networks," *University of Toronto*, vol. 415, 1994.

[86] V. Dutordoir, H. Salimbeni, E. Hambro, J. McLeod, F. Leibfried, A. Artemev, M. van der Wilk, J. Hensman, M. P. Deisenroth, and S. John, "GPflux: A library for deep Gaussian processes," *arXiv preprint arXiv:2104.05674*, 2021.

[87] C. W. Rosenbrock, K. Gubaev, A. V. Shapeev, L. B. Pártay, N. Bernstein, G. Csányi, and G. L. Hart, "Machine-learned interatomic potentials for alloys and alloy phase diagrams," *Npj Comput. Mater.*, vol. 7, no. 1, pp. 1–9, 2021.

[88] D. J. C. MacKay, *NATO ASI series F computer and systems sciences*, vol. 168. Springer, 1998.

[89] A. N. Kolmogorov and A. T. Bharucha-Reid, *Foundations of the Theory of Probability: Second English Edition.* Courier Dover Publications, 2018.

[90] N. J. Higham, *Analysis of the Cholesky decomposition of a semi-definite matrix.* Oxford University Press, 1990.

[91] A. Krishnamoorthy and D. Menon, "Matrix inversion using Cholesky decomposition," in *2013 signal processing: Algorithms, architectures, arrangements, and applications (SPA)*, pp. 70–72, IEEE, 2013.

[92] G. H. Golub and C. F. Van Loan, *Matrix Computations.* JHU press, 2013.

[93] J. W. Ng and M. P. Deisenroth, “Hierarchical Mixture-of-Experts Model for Large-Scale Gaussian Process Regression,” *arXiv preprint arXiv:1412.3078*, vol. 1050, p. 9, 2014.

[94] J. Broad, S. Preston, R. J. Wheatley, and R. S. Graham, “Gaussian process models of potential energy surfaces with boundary optimization,” *J. Chem. Phys.*, vol. 155, no. 14, p. 144106, 2021.

[95] M. Abramowitz, I. A. Stegun, and R. H. Romer, “Handbook of Mathematical Functions with Formulas, Graphs, and MathematicalTtables,” 1988.

[96] G. Schwarz, “Estimating the dimension of a model,” *Ann. Stat.*, pp. 461–464, 1978.

[97] D. Unruh, R. V. Meidanshahi, S. M. Goodnick, G. Csányi, and G. T. Zimányi, “Training a Machine-Learning Driven Gaussian Approximation Potential for Si-H Interactions,” *arXiv preprint arXiv:2106.02946*, 2021.

[98] E. Uteva, *Gaussian Process Machine Learning of Potential Energy Hypersurfaces.* PhD thesis, University of Nottingham, 2018.

[99] C. Møller and M. S. Plesset, “Note on an approximation treatment for many-electron systems,” *Phys. Rev.*, vol. 46, no. 7, p. 618, 1934.

[100] F. Coester, “Bound states of a many-particle system,” *Nucl. Phys.*, vol. 7, pp. 421–424, 1958.

[101] F. Coester and H. Kümmel, “Short-range correlations in nuclear wave functions,” *Nucl. Phys.*, vol. 17, pp. 477–485, 1960.

[102] J. Čížek and J. Paldus, “Correlation problems in atomic and molecular systems III. Rederivation of the coupled-pair many-electron theory using the traditional quantum chemical methodst,” *Int. J. Quantum Chem.*, vol. 5, no. 4, pp. 359–379, 1971.

[103] R. J. Bartlett and M. Musiał, “Coupled-cluster theory in quantum chemistry,” *Rev. Mod. Phys.*, vol. 79, no. 1, p. 291, 2007.

[104] D. J. C. MacKay, "Information-based objective functions for active data selection," *Neural Comput.*, vol. 4, no. 4, pp. 590–604, 1992.

[105] L. E. Atlas, D. A. Cohn, and R. E. Ladner, "Training connectionist networks with queries and selective sampling," in *Adv. Neural Inf. Process Syst.*, pp. 566–573, Citeseer, 1990.

[106] R. V. Krems, "Bayesian machine learning for quantum molecular dynamics," *Phys. Chem. Chem. Phys.*, vol. 21, no. 25, pp. 13392–13410, 2019.

[107] P. Valentini, M. S. Grover, N. Bisek, and A. Verhoff, "Molecular simulation of flows in thermochemical non-equilibrium around a cylinder using ab initio potential energy surfaces for N_2+N and N_2+N_2 interactions," *Phys. Fluids*, vol. 33, no. 9, p. 096108, 2021.

[108] A. D. Danilack, S. J. Klippenstein, Y. Georgievskii, and C. F. Goldsmith, "Low-temperature oxidation of diethyl ether: Reactions of hot radicals across coupled potential energy surfaces," *Proc. Combust. Inst.*, vol. 38, no. 1, pp. 671–679, 2021.

[109] J. C. Slater, "A simplification of the Hartree-Fock method," *Phys. Rev.*, vol. 81, no. 3, p. 385, 1951.

[110] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," *Phys. Rev.*, vol. 140, no. 4A, p. A1133, 1965.

[111] F. Jensen, *Introduction to Computational Chemistry.* John wiley & sons, 2017.

[112] M. Born and R. Oppenheimer, "On the Quantum Theory of Molecules," *Ann. Phys.*, vol. 389, no. 20, pp. 457–484, 1927.

[113] T. H. Dunning Jr, "Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen," *J. Chem. Phys.*, vol. 90, no. 2, pp. 1007–1023, 1989.

[114] K. A. Peterson and T. H. Dunning Jr, "Accurate correlation consistent basis sets for molecular core–valence correlation effects: The second row atoms

Al–Ar, and the first row atoms B–Ne revisited," *J. Chem. Phys.*, vol. 117, no. 23, pp. 10548–10560, 2002.

[115] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," *Phys. Rev.*, vol. 140, no. 4A, p. A1133, 1965.

[116] A. Stone, *The Theory of Intermolecular Forces.* OUP Oxford, 2013.

[117] J. N. Israelachvili, *Intermolecular and Surface Forces.* Academic press, 2011.

[118] D. Hankins, J. Moskowitz, and F. Stillinger, "Water molecule interactions," *J. Chem. Phys.*, vol. 53, no. 12, pp. 4544–4554, 1970.

[119] G. Alagona and C. Ghio, "*Ab initio* theoretical methods for studying intermolecular forces," 2003.

[120] S. F. Boys and F. Bernardi, "The calculation of small molecular interactions by the differences of separate total energies. Some procedures with reduced errors," *Mol. Phys.*, vol. 19, no. 4, pp. 553–566, 1970.

[121] D. Hankins, J. Moskowitz, and F. Stillinger, "Water molecule interactions," *J. Chem. Phys.*, vol. 53, no. 12, pp. 4544–4554, 1970.

[122] J. Del Bene and J. Pople, "Intermolecular energies of small water polymers," *Chem. Phys. Lett.*, vol. 4, no. 7, pp. 426–428, 1969.

[123] K. Kim, M. Dupuis, G. Lie, and E. Clementi, "Revisiting small clusters of water molecules," *Chem. Phys. Lett.*, vol. 131, no. 6, pp. 451–456, 1986.

[124] M. Riera, E. Lambros, T. T. Nguyen, A. W. Götz, and F. Paesani, "Low-order many-body interactions determine the local structure of liquid water," *Chem. Sci.*, vol. 10, no. 35, pp. 8211–8218, 2019.

[125] J. Salcedo-Gallo, L. Fallarino, J. Alzate-Cardona, E. Restrepo-Parra, and A. Berger, "Monte Carlo simulations of the thermodynamic behavior of exchange graded ferromagnets," *Phys. Rev. B*, vol. 103, no. 9, p. 094440, 2021.

[126] F. Moradi, K. R. E. Saraee, S. A. Sani, and D. Bradley, "Metallic nanoparticle radiosensitization: The role of Monte Carlo simulations towards progress," *Radiat. Phys. Chem.*, vol. 180, p. 109294, 2021.

[127] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids.* Oxford university press, 2017.

[128] R. J. Sadus, *Molecular Simulation of Fluids.* Elsevier, 2002.

[129] A. Z. Panagiotopoulos, "Direct determination of phase coexistence properties of fluids by Monte Carlo simulation in a new ensemble," *Mol. Phys.*, vol. 61, no. 4, pp. 813–826, 1987.

[130] A. Z. Panagiotopoulos, N. Quirke, M. Stapleton, and D. Tildesley, "Phase equilibria by simulation in the Gibbs ensemble: alternative derivation, generalization and application to mixture and membrane equilibria," *Mol. Phys.*, vol. 63, no. 4, pp. 527–545, 1988.

[131] K. Leonhard and U. Deiters, "Monte Carlo simulations of neon and argon using *ab initio* potentials," *Mol. Phys.*, vol. 98, no. 20, pp. 1603–1616, 2000.

[132] B. Axilrod and E. Teller, "Interaction of the van der Waals type between three atoms," *J. Chem. Phys.*, vol. 11, no. 6, pp. 299–300, 1943.

[133] GPy, "GPy: A Gaussian process framework in python." http://github.com/SheffieldML/GPy, since 2012.

[134] H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, and M. Schütz, "Molpro: a general-purpose quantum chemistry program package," *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, vol. 2, no. 2, pp. 242–253, 2012.

[135] H.-J. Werner and P. J. Knowles, "An efficient internally contracted multiconfiguration–reference configuration interaction method," *J. Chem. Phys.*, vol. 89, no. 9, pp. 5803–5814, 1988.

[136] P. J. Knowles and H.-J. Werner, "An efficient method for the evaluation of coupling coefficients in configuration interaction calculations," *Chem. Phys. Lett.*, vol. 145, no. 6, pp. 514–522, 1988.

[137] G. E. Scuseria, M. D. Miller, F. Jensen, and J. Geertsen, "The dipole moment of carbon monoxide," *J. Chem. Phys.*, vol. 94, no. 10, pp. 6660–6663, 1991.

[138] H.-J. Werner and P. Rosmus, "Theoretical dipole moment functions of the HF, HCl, and HBr molecules," *J. Chem. Phys.*, vol. 73, no. 5, pp. 2319–2328, 1980.

[139] C. Qu, Q. Yu, B. L. Van Hoozen Jr, J. M. Bowman, and R. A. Vargas-Hernández, "Assessing Gaussian process regression and permutationally invariant polynomial approaches to represent high-dimensional potential energy surfaces," *J. Chem. Theory Comput.*, vol. 14, no. 7, pp. 3381–3396, 2018.

[140] R. Kumar and J. L. Skinner, "Water simulation model with explicit three-molecule interactions," *J. Phys. Chem. B*, vol. 112, no. 28, pp. 8311–8318, 2008.

[141] J. M. Pedulla, F. Vila, and K. Jordan, "Binding energy of the ring form of H_2O_6: Comparison of the predictions of conventional and localized-orbital MP2 calculations," *J. Chem. Phys.*, vol. 105, no. 24, pp. 11091–11099, 1996.

[142] M. P. Hodges, A. J. Stone, and S. S. Xantheas, "Contribution of many-body terms to the energy for small water clusters: A comparison of *ab initio* calculations and accurate model potentials," *J. Phys. Chem. A*, vol. 101, no. 48, pp. 9163–9168, 1997.

[143] L. Ojamaee and K. Hermansson, "*Ab initio* study of cooperativity in water chains: Binding energies and anharmonic frequencies," *J. Phys. Chem.*, vol. 98, no. 16, pp. 4271–4282, 1994.

[144] A. E. Nasrabad, R. Laghaei, and U. Deiters, "Prediction of the thermophysical properties of pure neon, pure argon, and the binary mixtures neon-argon and argon-krypton by Monte Carlo simulation using *ab initio* potentials," *J. Chem. Phys.*, vol. 121, no. 13, pp. 6423–6434, 2004.

[145] R. Bukowski and K. Szalewicz, "Complete *ab initio* three-body nonadditive potential in Monte Carlo simulations of vapor-liquid equilibria and pure phases of argon," *J. Chem. Phys.*, vol. 114, no. 21, pp. 9518–9531, 2001.

[146] E. M. Mas, R. Bukowski, and K. Szalewicz, "*Ab initio* three-body interactions for water. II. Effects on structure and energetics of liquid," *J. Chem. Phys.*, vol. 118, no. 10, pp. 4404–4413, 2003.

[147] R. Bukowski, K. Szalewicz, G. C. Groenenboom, and A. van der Avoird, "Polarizable interaction potential for water from coupled cluster calculations. II. Applications to dimer spectra, virial coefficients, and simulations of liquid water," *J. Chem. Phys.*, vol. 128, no. 9, p. 094314, 2008.

[148] B. Widom, "Some topics in the theory of fluids," *J. Chem. Phys.*, vol. 39, no. 11, pp. 2808–2812, 1963.