ACEpotentials.jl

Witt, William C.; Oord, Cas van der; Gelžinytė, Elena; Järvinen, Teemu; Ross, Andres; Darby, James P.; Ho, Cheuk Hin; Baldwin, William J.; Sachs, Matthias; Kermode, James; Bernstein, Noam; Csányi, Gábor; Ortner, Christoph

DOI:
10.48550/arXiv.2309.03161

ACEpotentials.jl : A Julia Implementation of the Atomic Cluster Expansion

William C Witt,[1] Cas van der Oord,[2] Elena Gelžinytė,[2] Teemu Järvinen,[3] Andres Ross,[3] James P. Darby,[4] Cheuk Hin Ho,[3] William J. Baldwin,[2] Matthias Sachs,[5] James Kermode,[4] Noam Bernstein,[6] Gábor Csányi,[2,*] and Christoph Ortner[3,†]

[1] *Department of Materials Science & Metallurgy, University of Cambridge, Cambridge, United Kingdom*
[2] *Engineering Laboratory, University of Cambridge, Cambridge, CB2 1PZ UK*
[3] *Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, BC, Canada V6T 1Z2*
[4] *Warwick Centre for Predictive Modelling, School of Engineering, University of Warwick, Coventry, CV4 7AL, United Kingdom*
[5] *School of Mathematics, University of Birmingham, Birmingham, B15 2TT, United Kingdom*
[6] *Center for Materials Physics and Technology, U. S. Naval Research Laboratory, Washington, DC, 20375, United States of America*

(Dated: September 8, 2023)

We introduce `ACEpotentials.jl`, a Julia-language software package that constructs interatomic potentials from quantum mechanical reference data using the Atomic Cluster Expansion *(Drautz, 2019)*. As the latter provides a complete description of atomic environments, including invariance to overall translation and rotation as well as permutation of like atoms, the resulting potentials are systematically improvable and data efficient. Furthermore, the descriptor's expressiveness enables use of a linear model, facilitating rapid evaluation and straightforward application of Bayesian techniques for active learning. We summarize the capabilities of `ACEpotentials.jl` and demonstrate its strengths (simplicity, interpretability, robustness, performance) on a selection of prototypical atomistic modelling workflows.

I. INTRODUCTION

Machine-learning interatomic potentials (MLIPs) continue to revolutionize the fields of molecular and materials simulation [1, 2]. MLIPs provide the means to simulate atomistic systems at or close to the accuracy of electronic structure methods, while being computationally cheaper by orders of magnitude. They make the simulation of large-scale systems and long time-scales at high model accuracy accessible and have therefore become an indispensable tool for atomic-scale simulation. Recent reviews of the the field are provided in [3–6]. Of particular relevance to the present work are the methods introduced in [2, 7–9].

To create an MLIP, one begins with a flexible functional form, constrained only to comply with the natural symmetries of the potential energy in three-dimensional space, then estimates its parameters using reference data, typically in the form of energies, forces, and virial stresses for a set of representative atomic configurations. Ordinarily, the data are generated with quantum mechanical techniques, such as density functional theory calculations, which may be performed only for relatively small structures. A well-trained MLIP is then expected to provide accurate predictions of processes on similar but also much larger spatial scales.

The *Atomic Cluster Expansion* (ACE) introduced in [9] is a particular MLIP flavor that is flexible, theoretically well founded, interpretable, and for which it is straightforward to tune the cost-accuracy balance. It is establishing itself as a successful MLIP approach for a wide range of tasks, especially but not exclusively in materials simulation; see e.g. [10–17]. Linear variants of the ACE model have been found remarkably data efficient and computationally efficient and as such have proven particularly useful for active learning (AL) workflows [14] as Sec. III and Sec. IV will demonstrate. Linearity in particular enables sensitivity analysis and a path towards reliable uncertainty quantification.

This article describes `ACEpotentials.jl`, which ties together a collection of Julia-language packages to expose a user-oriented interface facilitating the convenient construction of ACE MLIPs. To highlight the ease of use of our package, Listing 1 provides a complete Julia-language example that produces an ACE potential for a TiAl dataset.

* Corresponding author, gc121@cam.ac.uk
† Corresponding author, ortner@math.ubc.ca

```julia
using ACEpotentials
data, _, _ = ACEpotentials.example_dataset("TiAl_tutorial")
model = acemodel(elements = [:Ti, :Al],
                 order = 3,
                 totaldegree = 12,
                 Eref = [:Ti => -1586.0195, :Al => -105.5954])
acefit!(model, data)
export2lammps("TiAl_tutorial.yace", model)
```

Listing 1: A minimal Julia-language script for fitting an `ACEpotentials.jl` potential. It first downloads a training dataset, then uses `acemodel` to create a model object whose parameters are explained fully in the following sections. The model parameters are estimated with the `acefit!` command, and the result is exported in a LAMMPS compatible format.

At the time of writing, `ACEpotentials.jl` provides interfaces for *linear* ACE models, which give good accuracy as well as performance both in parameter estimation and prediction. We have incorporated a range of geometric and analytical priors into the default model parameters that have proven robust in a range of tasks, including the challenging low data regime arising in active learning workflows. `ACEpotentials.jl` models can be used for molecular dynamics simulation in LAMMPS [18], ASE [19] and `Molly.jl` [20].

The Julia-language codes on which `ACEpotentials.jl` builds are written with ease-of-use, performance, and flexibility of model development in mind. Several variations and extensions of the ACE model implementations discussed in this article are under active development. The choice of Julia as the development language enables seamless transition from rapid prototyping to performance optimization. Moreover, Julia is establishing itself as leader in *scientific machine learning* (see, e.g., [21]), facilitating highly customized model architectures with novel computational kernels.

Finally, we emphasize that the aim of this article is to illustrate the capabilities of `ACEpotentials.jl` but not to precisely document its use; for the latter see the reference material at [22], which will evolve along with the software. While the examples and code snippets provided throughout this article are compatible with the present version of `ACEpotentials`.jl, they should be taken primarily as illustrations of how the package may be used. The documentation will be kept up-to-date for the foreseeable future and will continually expand to describe additional options and features.

II. METHODS

A. Review of the linear ACE framework

Model Specification

An atomic structure is described by a collection of position-element pairs $(\boldsymbol{r}_i, Z_i)$, and the computational unit cell (with open or periodic boundary conditions). In the ACE model, the total potential energy of such a structure is decomposed into site energies,

$$E = \sum_i \varepsilon_i, \tag{1}$$

where the summation ranges over all atoms belonging to the computational cell and each ε_i depends on its atomic neighbourhood containing all atoms within a cutoff radius $r_{\rm cut}$ from $\boldsymbol{r}_i$, taking into account the boundary conditions. The ACE framework provides a design space to construct systematic models for the site energy ε_i in terms of a complete linear basis of body-ordered symmetric polynomials.

For convenience we introduce the new variables $\boldsymbol{x}_i := (\boldsymbol{r}_i, Z_i)$ for the state of an atom and $\boldsymbol{x}_{ij} := (\boldsymbol{r}_{ij}, Z_i, Z_j)$, where $\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$, for the state of a bond between atoms $\boldsymbol{x}_i, \boldsymbol{x}_j$. In terms of these variables the site energy is expanded in body-order, in two different formulations:

$$\varepsilon_i = V^{(0)}(Z_i) + \sum_{j_1} V^{(1)}(\boldsymbol{x}_{ij_1}) + \sum_{j_1<j_2} V^{(2)}(\boldsymbol{x}_{ij_1}, \boldsymbol{x}_{ij_2}) + \cdots + \sum_{j_1<\cdots<j_{\bar\nu}} V^{(\bar\nu)}(\boldsymbol{x}_{ij_1}, \ldots, \boldsymbol{x}_{ij_{\bar\nu}}) \tag{2a}$$

$$= V^{(0)}(Z_i) + \sum_{j_1} U^{(1)}(\boldsymbol{x}_{ij_1}) + \frac{1}{2!}\sum_{j_1,j_2} U^{(2)}(\boldsymbol{x}_{ij_1}, \boldsymbol{x}_{ij_2}) + \cdots + \frac{1}{\bar\nu!}\sum_{j_1,\ldots,j_{\bar\nu}} U^{(\bar\nu)}(\boldsymbol{x}_{ij_1}, \ldots, \boldsymbol{x}_{ij_{\bar\nu}}). \tag{2b}$$

We call the first formulation (2a) the *canonical cluster expansion*. It can be transformed [9] into the second formulation (2b), where the sums run over all possible combinations of atoms, including all permutation-equivalent clusters and even "artificial clusters" with repeated particles. This transformation introduces unphysical self-interaction terms such as $V^{(2)}(\boldsymbol{x}_{ij}, \boldsymbol{x}_{ij})$, but this counter-intuitive choice leads to a tensor product structure that can be exploited in constructing a highly efficient evaluation scheme. Our code is unique in that it implements the transformation between the two descriptions and also allows the evaluation of the canonical formulation (2a). Indeed the default `ACEpotentials.jl` model specification uses a combination of the two formulations. We will briefly review the challenges involved in evaluating cluster expansion models in Appendix A.

Both series in (2) are truncated versions of an exact body-order expansion. An exact expansion would include terms up to the number of atoms in the system, while here the maximum body-order is $\bar{\nu}+1$ (corresponding to a correlation order of $\bar{\nu}$), which constitutes the first approximation parameter. In practice, the truncation is performed at low to moderate $\bar{\nu}$ (typically 5 or less) for several reasons, including control of model complexity and computational cost.

Each potential $V^{(\nu)}$ (or, $U^{(\nu)}$) is parameterized by a linear model, a process for which we give details below in the following sections. This then results in a parameterisation of the site energy that is also linear,

$$\varepsilon_i = \boldsymbol{c} \cdot \boldsymbol{B}_i, \tag{3}$$

where $\boldsymbol{c}$ is a vector of parameters and $\boldsymbol{B}_i$ a vector of basis functions (or, features) involved in the expansion of the many-body potentials $V^{(\nu)}$ or $U^{(\nu)}$. The basis functions are by construction invariant under rotations, reflections and permutations of like atoms. The representation is also *complete* (or, universal) in the sense that when the approximation parameters (body-order, cutoff radius, and expansion resolution) are taken to infinity, the model can in principle represent an arbitrary smooth site-energy potential. Linearity of the model allows us to employ a vast range of established tools for parameter estimation and uncertainty quantification, and enables rapid model development by refitting to new training data or with adjusted hyperparameters.

The basis functions $\boldsymbol{B}_i$ specify the model. In a typical example this can be done as demonstrated in Listing 2.

```
1  using ACEpotentials
2  model = acemodel(; elements = [:Ti, :Al],
3                     order = 3,
4                     totaldegree = 12,
5                     rcut = 5.5,
6                     Eref = [:Ti => -1586.0195, :Al => -105.5954])
```

`elements`	list of chemical elements occurring in the system of interest
`order`	maximum correlation order, $\bar{\nu}$ in the article text; cf. Eq. (2)
`totaldegree`	spatial resolution of the ν-body potentials; cf. Eq. (9)
`rcut`	(optional) cutoff radius; cf. Sec. II B
`Eref`	(optional) reference energies specifying $V^{(0)}(Z_i)$

Listing 2: A typical construction of an ACE model and description of parameters.

The `model` object specifies the model site energy potential, from which derived properties such as potential energy, forces and virial stresses can be computed that are used in molecular statics, molecular dynamics or sampling algorithms.

There are many additional parameters and options available to specify an ACE model, some of which we discuss throughout the remainder of this paper. For a complete list of options we refer to the documentation [22]. We only remark briefly on the `Eref` parameter: We recommend the explicit specification of the one-body term $V^{(0)}$. We observed in many tests that constraining $V^{(0)}(Z_i)$ to be the energy of a single isolated atom with atomic number Z_i yields more chemically realistic potentials that are more robust in practical molecular dynamics and molecular statics simulations, especially those involving breaking and forming bonds. One provides this information to an ACE model as shown in Listing 2, line 6.

In the remainder of this section we maintain a focus on high level intuitive understanding of options and parameters and avoid details and technicalities of the ACE framework as much as possible. For those details we refer to Appendix A and to the many publications now available on the subject [6, 9, 11, 12].

Parameter Estimation

Having specified a physically reasonable model architecture, we must now estimate its parameters. To that end we require a training set, which typically consists of a list of atomic structures, $\boldsymbol{R} = \{R\}$, for which the total potential

energy $\mathscr{E}_R \in \mathbb{R}$, forces $\mathscr{F}_R \in \mathbb{R}^{3\times N_R}$ (with N_R the number of atoms in the computational unit cell) and possibly also virial stresses $\mathscr{V}_R \in \mathbb{R}^6$ (in Voigt notation) have been evaluated with an electronic structure model. We define $E(\boldsymbol{c}; R), F(\boldsymbol{c}; R), V(\boldsymbol{c}; R)$ be the corresponding energies, forces and virials for the structure R in the ACE model, with parameters $\boldsymbol{c}$. The simplest way to estimate those parameters is then to minimize the least squares loss function

$$L(\boldsymbol{c}) = \sum_{R\in\boldsymbol{R}} \Big(w_{E,R}^2 |E(\boldsymbol{c}; R) - \mathscr{E}_R|^2 + w_{F,R}^2 |F(\boldsymbol{c}; R) - \mathscr{F}_R|^2 + w_{V,R}^2 |V(\boldsymbol{c}; R) - \mathscr{V}_R|^2 \Big). \tag{4}$$

The weights $w_{E,R}, w_{F,R}, w_{V,R}$ can be used to give more or less relative "importance" to certain structures or observations. They are usually highly structured (e.g., $w_{E,R}, w_{V,R}$ are scaled with the number of atoms in a structure R), which will be discussed in more detail in Section II E. Since the ACE model is linear in $\boldsymbol{c}$ it follows that $L(\boldsymbol{c})$ is quadratic, which means that minimizing L is a linear least squares problem. A wide range of efficient numerical techniques exist for its solution. In particular we will normally employ regularized or Bayesian variations of the naive least squares minimization, which are discussed in Sections II E and II F.

In Listing 3 we read in such a prepared training set provided in the extended XYZ format and then estimate the model parameters with a default solver (Bayesian Linear Regression; cf. Section II F). Several steps are combined and hidden from the user in the `acefit!` convenience function, but all these steps can in principle also be performed manually, e.g., to explore different parameter estimation algorithms that are currently not interfaced by `ACEpotentials.jl`. In line 5 of the listing, the fitted model is exported to a format that can be used for molecular dynamics simulations in LAMMPS.

```
1 model = ... # cf. Listing 2
2 P = smoothness_prior(model)
3 data, _, _ = ACEpotentials.example_dataset("TiAl_tutorial")
4 acefit!(model, data; prior = P, solver = ACEfit.BLR())
5 export2lammps("TiAl.yace", model)
```

`smoothness_prior`	specifies a model prior / regularizer; cf. Section II C
`pathtodata`	absolute path to a small training set used for testing
`data`	collection of structures containing training data; cf. Section II D
`acefit!`	assembles and solves the least squares system; cf. Section II E
`ACEfit.BLR()`	default solver for parameter estimation; cf. Section II F
`export2lammps`	exports the model to a LAMMPs readable format.

Listing 3: A representative example loading a training dataset and estimating ACE model parameters.

In the remainder of Section II we will dive slightly deeper into some the steps we outlined above. Then, in Section III we will demonstrate how the framework can be used to fit potential energy models for realistic materials and molecular systems of scientific interest.

B. Choice of basis functions & Geometric priors

The parameters in the model specification in Listing 2 specify a basis in which the $V^{(\nu)}$ potentials are expanded. In the current section we will detail the *basis functions* that are employed, while in Section II C we will then explain how to select a finite subset from the infinite complete basis set.

One-particle basis

To begin we must select a *one-particle basis* ϕ_k in which all smooth functions $f(\boldsymbol{x}_{ij}) = f(\boldsymbol{r}_{ij}, Z_i, Z_j)$ can be expanded. The most general form we consider is

$$\phi_{znlm}(\boldsymbol{r}_{ij}, Z_i, Z_j) = R_{nl}(r_{ij}, Z_i, Z_j) Y_l^m(\hat{\boldsymbol{r}}_{ij}) \delta_{zZ_j}, \tag{5}$$

where δ denotes the Kronecker symbol and we have identified $k = (z, n, l, m)$. The Y_l^m are the standard complex spherical harmonics, while R_{nl} is called the *radial basis*. The choice of Y_l^m to embed the angular component $\hat{\boldsymbol{r}}_{ij}$ facilitates the exact symmetrization of the parameterisation with respect to rotations. Since (r_{ij}, Z_i, Z_j) is already invariant under rotations, the choice of R_{nl} is extremely general. Nevertheless we will below outline a heuristic that

leads to a narrow class of choices that have proven successful in many applications. However, we note that the optimal choice of R_{nl} remains an active area of research and will likely also evolve within `ACEpotentials.jl`.

Once ϕ_k is selected, each potential $V^{(\nu)}$ (or, $U^{(\nu)}$) is expanded in terms of a tensor product many-body basis,

$$\begin{aligned} V^{(1)}\big(\boldsymbol{x}_{ij_1}\big) &= \sum_{k_1} c^{(Z_i)}_{k_1} \phi_{k_1}(\boldsymbol{x}_{ij_1}) \\ V^{(2)}\big(\boldsymbol{x}_{ij_1}, \boldsymbol{x}_{ij_2}\big) &= \sum_{k_1,k_2} c^{(Z_i)}_{k_1 k_2} \phi_{k_1}\big(\boldsymbol{x}_{ij_1}\big)\phi_{k_2}\big(\boldsymbol{x}_{ij_2}\big) \\ &\vdots \qquad\qquad \vdots \\ V^{(\bar\nu)}\big(\boldsymbol{x}_{ij_1}, \ldots, \boldsymbol{x}_{ij_{\bar\nu}}\big) &= \sum_{k_1,\ldots,k_{\bar\nu}} c^{(Z_i)}_{k_1\cdots k_{\bar\nu}} \phi_{k_1}\big(\boldsymbol{x}_{ij_1}\big)\cdots\phi_{k_{\bar\nu}}\big(\boldsymbol{x}_{ij_{\bar\nu}}\big) \end{aligned} \tag{6}$$

The model parameters $c^{(Z_i)}_{k_1\cdots k_\nu}$ will be estimated from data. Note that we choose individual model parameters for each center-atom element Z_i. During the parameter estimation, the parameters will be constrained to guarantee invariance of the resulting potentials under rotations and reflections of an atomic environment. Invariance under permutations is already ensured through the summation in (2). Appendix A reviews additional details of this invariant basis construction, resulting in the specification of $\boldsymbol{B}_i$ in terms of which site energy is defined in (3).

To complete the model specification two steps remain: (i) the choice of radial basis R_{nl}; and (ii) the selection of basis functions $(k_1, \ldots, k_\nu)$ that we employ in the expansions (6). In the remainder of this section we discuss (i) while (ii) will be discussed in Section II C.

Radial basis

There is considerable freedom in the choice of the radial basis R_{nl}, which can be thought of as a *geometric prior*. For example, it incorporates the interaction range (cutoff radius, $r_{\rm cut}$) and can be tuned to capture rough qualitative information about interacting atoms. In the following we describe a class of radial bases, available through `ACEpotentials.jl`, that require no data-driven optimization and thus leads to genuinely linear models. At the time of writing this article, `ACEpotentials.jl` supports radial bases indexed by n only, i.e. $R_{nl} = R_n$ for all l. This class is described by

$$R_n(r_{ij}, Z_j, Z_i) = f_{\rm env}(r_{ij}, Z_j, Z_i) P_n\big(y(r_{ij}, Z_j, Z_i)\big), \tag{7}$$

with the following components:

- y is an element-dependent distance transform, which can be used to impose increased spatial resolution where needed, especially near the equilibrium bond-length. We typically employ

  $$y(r_{ij}, Z_i, Z_j) = \left(1 + a\frac{(r/r_0)^q}{1 + (r/r_0)^{q-p}}\right)^{-1},$$

  where r_0 is an estimate of the equilibrium bond-length in the system and a is chosen to maximize the gradient of y at $r = r_0$, thereby maximizing resolution for nearest-neighbour interaction. The idea behind this transform is that it behaves as r^{-q} for large r and as $1 - r^p/a$ for small r thereby decreasing resolution in those two limits at rates determined by the parameters p, q. The reduction in resolution in the small r regime is desirable when no data is available to specify the model in that regime; see also Figure 1.

- P_n is an orthogonal basis in y-coordinates. Our default choice is the Legendre orthogonal polynomial basis, which implicitly assumes equidistribution of resolution in y-coordinates.

- Finally, $f_{\rm env}$ is an envelope that specifies the cutoff radius $r_{\rm cut}$.

  – The default and canonical choice for the many-body basis is

  $$f_{\rm env}(r_{ij}, Z_i, Z_j) = y^2(y - y_{\rm cut})^2,$$

  where $y_{\rm cut} = y(r_{\rm cut}, Z_i, Z_j)$.

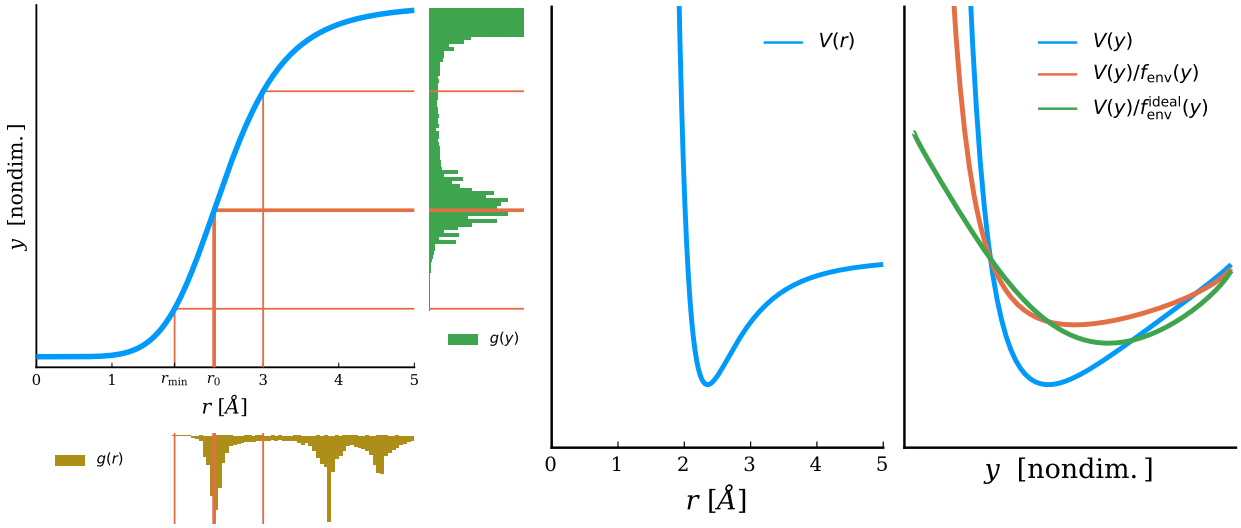


FIG. 1. **Center:** a typical interaction potential $V(r)$, plotted in r-coordinates. **Left:** a coordinate transform $y = y(r)$ to a non-dimensional variable y that increases resolution near $r = r_0$ where the potential minimum is located and decreases resolution below $r_{\min}$ (the radial distance occuring in the training dataset), to zero near $r = 0$ where there is no data (and the envelope f_{env} becomes relevant) and near $r = r_{\text{cut}}$ where the potential converges to a constant. The histograms show the distribution of a typical dataset in both r- and y-coordinates. **Right:** the interaction potential plotted (i) in transformed coordinates $V(r(y))$, (ii) with the default pair envelope removed and (iii) with the theoretically optimal, typically unknown, envelope removed. The parameterisation and the smoothness priors are not applied to the original potential $V(r)$ but to the transformed potential $V(y)/f_{\text{env}}(y)$.

– The default choice of envelope for the pair potential $U^{(1)}$ or $V^{(1)}$ is Coulomb potential tilted to ensure a smooth cutoff,

$$f_{\text{env}}(r_{ij}, Z_i, Z_j) = \left(\tfrac{r_{ij}}{r_0}\right)^{-1} - \left(\tfrac{r_{\text{cut}}}{r_0}\right)^{-1} + \left(\tfrac{r_{\text{cut}}}{r_0}\right)^{-2}\left(\tfrac{r_{ij}}{r_0} - \tfrac{r_{\text{cut}}}{r_0}\right),$$

which is repulsive as r_{ij}^{-1} as $r \to 0$ but continuously differentiable at the cutoff.

While the envelope for the many-body potential is canonical, for the pair potential envelope there is significant scope for inserting prior modelling knowledge of the system of interest. For example, one could replace the r^{-1} type behaviour with $r^{-p} + r^{-q}$ to obtain different behaviour as $r \to 0$ and $r \to r_{\text{cut}}$, or in fact one could incorporate the ZBL potential [23] to obtain asymptotically exact repulsion.

The effect of the distance transform $y = y(r)$ and of the envelope function are visualized in Figure 1.

- Repulsion restraint: The construction outlined above means that, in the canonical cluster expansion formulation, the pair potential is given by

$$V^{(1)}(r_{ij}, Z_i, Z_j) = f_{\text{env}}(r_{ij}, Z_i, Z_j) p_{Z_i Z_j}(y_{ij}),$$

where $p_{Z_i Z_j}$ is a polynomial in transformed y coordinates. By imposing the constraint that $p_{Z_i Z_j}(y_0) = 1$, where $y_0 = y(0, Z_i, Z_j)$, we ensure that $E \sim f_{\text{env}}(r_{ij})$ as $r_{ij} \to 0$. This *guarantees* repulsive behaviour of the total energy, independently of whether or not this is provided through the training data. In practice we enforce this weakly through a mild restraint to give the potential more flexibility.

```
using ACEpotentials
elements = [:Ti, :Al]
totaldegree = 12
r0 = (rnn(:Ti) + rnn(:Al)) / 2
rcut = 2 * r0
trans = AgnesiTransform(; r0=r0, p = 2)
fenv = PolyEnvelope(1, r0, rcut)
radbasis = transformed_jacobi_env(totaldegree, trans, fenv, rcut)
model = acemodel(elements = elements,
                 order = 3,
                 totaldegree = totaldegree,
                 radbasis = radbasis)
```

Listing 4: A example demonstrating more fine-grained control over the choice of radial basis R_{nl}. The function `transformed_jacobi_env` constructs the polynomial basis from which the radial basis is constructed, which can be within the general class of Jacobi polynomials, but is normally taken to be the Legendre basis in transformed y coordinates.

C. A priori sparsification & Smoothness prior

We now turn towards the second aspect of basis construction: how to select which of the infinitely many tensor product basis functions

$$\phi_{k_1} \otimes \cdots \otimes \phi_{k_\nu}, \tag{8}$$

specified by the tuples $(k_1, \ldots, k_\nu)$, we wish to incorporate into the expansion of the $(\nu+1)$-body potential $V^{(\nu)}$.

Sparse basis selection

Recall that $k_t = (z_t, n_t, l_t, m_t)$, and that the bound $|m_t| \le l_t$ on m_t automatically gives a selection of possible m_t values once l_t bounds are chosen. Roughly speaking, n_t, l_t measure how oscillatory the corresponding basis functions are in, respectively, the radial r_t and angular $\hat{\boldsymbol{r}}_t$ coordinates. Therefore one typically puts upper bounds $n_t \le n_{\max}$ and $l_t \le l_{\max}$ in the basis selection, i.e. one chooses all basis functions $(k_1, \ldots, k_\nu)$ in the expansion for which these bounds are satisfied. Lower bounds lead to a smaller basis, but also less flexibility and correspondingly lower accuracy on the training set.

This simple strategy is available in `ACEpotentials.jl` but the default usage takes the notion of regularity a step further and bounds the *mixed regularity* of the basis functions we select. This is done by choosing a maximum *total* degree totaldegree(ν) for each correlation order ν and choosing all basis functions $(k_1, \ldots, k_\nu)$ such that

$$1 \le \nu \le \bar{\nu} \quad \text{and} \quad \sum_{t=1}^{\nu} n_t + w_{\mathrm{L}} l_t \le \text{totaldegree}(\nu). \tag{9}$$

The additional weight w_{L} allows us to select whether we require lower or higher resolution of the angular versus radial components of the interaction. Note that a higher weight w_{L} decreases the angular resolution. The resulting selected basis is much sparser and is appropriate for parameterising very smooth functions in high dimension.

The default usage is that totaldegree(ν) takes the same value for all ν but one may also specify a separate total degree for each correlation order ν. For example, Listing 5 demonstrates how to select a stronger weight $w_{\mathrm{L}} = 2.0$ thus providing less angular resolution, as well as how to select total polynomial degrees $25, 23, 20, 10$ for, respectively, parameterising $V^{(1)}, V^{(2)}, V^{(3)}, V^{(4)}$.

```
using ACEpotentials
model = acemodel(elements = [:Ti, :Al],
                 order = 4,
                 wL = 2.0,
                 totaldegree = [25, 23, 20, 10])
```

`wL`	specifies the relative resolution in angular and radial basis
`totaldegree`	specify seperate degrees for each correlation order

Listing 5: Construct an ACE model with finer control on the sparse selection of basis functions.

Significant further fine-tuning of the basis specification is possible, e.g. choosing different total degrees and w_{L} parameters for different interacting species. This is explained in the package documentation [22].

Smoothness Prior

The foregoing discussion concludes the model *architecture* specification. An issue closely related to the sparse basis selection (9) is the definition of a smoothness prior that may be employed for ridge regression (regularized least squares) which we discuss in Section II E or in the Bayesian framework of Section II F. As explained above, the value

$$\sum_{t=1}^{\nu} n_t + w_{\mathrm{L}} l_t$$

is a qualitative estimate for how oscillatory or smooth a basis function (8) is. We can extend this definition slightly by adding another parameter p and defining

$$\gamma_{\boldsymbol{znlm}} := \sum_{t=1}^{\nu} n_t^p + w_{\mathrm{L}} l_t^p, \tag{10}$$

where $\boldsymbol{z} = (z_t)_{t=1}^{\nu}, \boldsymbol{n} = (n_t)_{t=1}^{\nu}, \boldsymbol{l} = (l_t)_{t=1}^{\nu}$ and $\boldsymbol{m} = (m_t)_{t=1}^{\nu}$. We then collect these parameters into a diagonal matrix Γ with $\Gamma_{\boldsymbol{kk}} = \gamma_{\boldsymbol{k}}$. If $\boldsymbol{c}$ are the model parameters then $\|\Gamma \boldsymbol{c}\|_2$ will be a rough estimate for how smooth the potential energy surface is.

The matrix Γ also serves as a smoothness prior within the Bayesian interpretation of ridge regression: the prior distribution for the model parameters $\boldsymbol{c}$ is given by a multivariate normal distribution that is centered at the origin and has variance proportional to Γ^{-2}; see Sections II E and II F. In `ACEpotentials.jl` this operator can be constructed as shown in Listing 6, with $p = 4, w_{\mathrm{L}} = 1$ the default.

```
model = ... # cf. Listing 2
Γ = smoothness_prior(model; p = 4, wL = 1)
```

Listing 6: Construct an operator that estimates the smoothness of the MLIP model, to be used as a Tikhonov regulariser, or prior in a Bayesian framework.

The resulting operator Γ may now be used to specify the regularizer (or prior) of parameter estimation algorithms, e.g., in Listing 3, line 2 and explained in more detail in Sections II E and II F. A key point is that Γ is a *rigorous* smoothness prior for the canonical cluster expansion (2a) but only a heuristic for the self-interacting expansion (2b).

It is interesting in general, but in particular in the low-data regime, to explore different choices of priors. Two particular variants that are also available in `ACEpotentials.jl` are the exponential and Gaussian priors

$$\gamma^{\mathrm{exp}}_{\boldsymbol{znlm}} = \exp\Big(\alpha_{\mathrm{l}} \sum_t l_t + \alpha_{\mathrm{n}} \sum_t n_t\Big), \qquad \text{and} \qquad \gamma^{\mathrm{gauss}}_{\boldsymbol{znlm}} = \exp\Big(\sigma_{\mathrm{l}} \sum_t l_t^2 + \sigma_{\mathrm{n}} \sum_t n_t^2\Big),$$

which enforce even stronger smoothness requirements than the algebraic prior (10) and are currently still experimental features.

D. Training data

In the foregoing sections we discussed in some depth how an ACE interatomic potential architecture can be conveniently specified. The next task is to estimate the parameters matching the model to training data.

A training dataset consists of a collection of reference structures, $\boldsymbol{R} = \{R\}$, each with associated potential energy $\mathscr{E}_R \in \mathbb{R}$, forces $\mathscr{F}_R \in \mathbb{R}^{3\times N_R}$ and, when appropriate, virials $\mathscr{V}_R \in \mathbb{R}^6$ (Voigt notation). The reference energies, forces and virials are typically obtained by evaluating a "high fidelity" reference potential energy surface for which we wish to obtain an ACE surrogate model. Density Functional Theory is a common choice, but higher levels of theory such as Coupled-Cluster methods are also used, especially for non-periodic systems. In addition each training structure should be given a label that specifies related sub-groups. For example, these subgroups could indicate different phases of a material, and the resulting labels might be "bcc", "fcc", "liquid". The label could also indicate the MD temperature from the which the structures were generated, e.g. "fcc500K" or "liquid2500K". This allows convenient filtering of the training set, e.g., for assigning training weights (cf. Section II E) or fitting to subsets.

Acquisition of training data need not be performed within the `ACEpotentials.jl` package, but can be undertaken in any simulation software that makes it convenient to generate and manipulate atomic structures, perform molecular dynamics or Monte Carlo simulations, and to evaluate structures using a high fidelity electronic structure model. Because of the general ease of use and in particular ease of interoperability with the Julia molecular simulation eco-system, we often use the Atomic Simulation Environment [24].

The standard format for storing and retrieving a training set in `ACEpotentials.jl` is the extended XYZ format and can be read as shown in Listing 7. This results in a list of atomic structures storing the structure information as well as the training data.

```
using ACEpotentials
pathtodata = "path/to/data.xyz"
data = read_extxyz(pathtodata)
```

Listing 7: Reading a training set from an extended XYZ file.

Overview of Training Set Acquisition

The acquisition of training data is often the most time-consuming aspect of MLIP development. An in-depth discussion goes beyond the scope of this software review article; important details can be found for example in [5, 12, 25–27]. In the remainder of this section we give an outline of general strategies to consider, while in Section III we go into practical aspects how training sets can be constructed in a few prototypical applications and what kind of tools `ACEpotentials.jl` provide to support that task.

The overarching requirements are that training sets (1) must contain small enough atomic structures that they can be evaluated using high-fidelity electronic structure models; and (2) must contain snapshots of all possible local atomic configurations one expects to encounter during simulation and prediction tasks. Thus, generating a training set reduces to generating representative atomic structures which are then evaluated with the reference model to obtain target potential energies, forces and virials. While the latter is usually straightforward and varies little between projects, there is no standard way yet to generate the training structures. The choice will depend on the atomic system at hand, and the simulation tasks that the model must be able to perform reliably, e.g. which system properties (observables) are to be modelled.

As a first step, one should "sketch out" the parts of the potential energy landscape that are of interest, e.g. construct one representative structure for each distinct energy minimum of interest. This might include different phases or material defects that the final model should be able to describe. Next, one generates random samples from those sketches for example by displacing the atom positions (randomly, along normal modes, volume scans, and so forth), or by subsampling an *ab initio* molecular dynamics trajectory. After collecting a seemingly adequate number of training structures (the total number of observations should normally exceed that number of parameters) one can fit a first model and test that model's accuracy with respect to some target property. If the accuracy is inadequate, or the model not robust (e.g., an MD simulation is unstable), then a good strategy is to proceed with an iterative model refinement process. In each iteration additional training structures are selected to converge the model's accuracy with respect to the target properties of interest. One might add hand-crafted structures to fix a particular flaw (e.g. to improve description of inter-molecular interaction in a molecular liquid or include supercells with vacant atomic sites) or model-driven MD to less computationally expensively explore relevant parts of Potential Energy Surface (for example, low potential energy regions to bring potential-Boltzmann-sample closer to reference-Boltzmann-sample

and wider temperature/pressure range than intended for application of interest to make the model-driven simulations more stable).

Iterative model refinement is closely related to *active learning*. That strategy assumes that there is an accurate and efficient way available to estimate model uncertainty. During a simulation task, for example a molecular dynamics simulation, when a structure with high uncertainty is encountered it is evaluated with a reference method and added to the training data. To accelerate this process, we developed Hyper-Active Learning [14], which biases molecular dynamics simulation towards high-uncertainty and high predicted error regions. This strategy is sometimes capable of more rapidly generating many independent training samples. Section III will go into some details how this strategy is used in practice.

E. Parameter estimation: ridge regression

Recall from Section II A that the linear ACE models are parameterized linearly as shown in (3). As described in Section II D we estimate parameters by matching the model to observations of total energies, forces and virials evaluated via a high fidelity reference model on different training structures $R \in \boldsymbol{R}$, where $\boldsymbol{R}$ denotes the training set. To estimate the parameters we minimize the loss function (4). In the current section, we go into further details of the parameter estimation process once the model and training set have been specified.

First, we discuss the regression weights $w_{E,R}$, $w_{F,R}$ and $w_{V,R}$, which allow users to specify the relative importance of different observations and structures. In principle one could specify individual weights for each structure R and observation type E, F, V. In practice, it has proven convenient to label all structures R with a *configuration type* as described in Section II D and to assign weights according to such groups. In addition the weights $w_{E,R}, w_{V,R}$ should scale like $1/\sqrt{N_R}$ where N_R denotes the number of atoms in the structure R [2, 12]. Thus, the weights $w_{E,R}, w_{V,R}$ take the form

$$w_{E,R} = \frac{\tilde{w}_{E,\mathrm{cfgtype}(R)}}{\sqrt{N_R}}, \qquad w_{F,R} = \tilde{w}_{F,\mathrm{cfgtype(R)}}, \qquad w_{V,R} = \frac{\tilde{w}_{V,\mathrm{cfgtype}(R)}}{\sqrt{N_R}},$$

with $\tilde{w}_{*,\mathrm{cfgtype}}$ defined by the user as follows: Suppose, for example, that a training set contains several solid phase structures as well as liquid structures, then we may wish to demand a higher fit accuracy on the solid structures. In addition we typically find that energies must be given higher weights in order to achieve the best possible balance of accuracy. This might result in weight specifications as shown in Listing 8, lines 4-5.

```
1  model = ... # specify a model; see e.g. Listing 2
2  data = ...  # load training data; see e.g. Listing 7
3  P = smoothness_prior(model)        # regularisation operator; see § II C
4  weights = Dict( "default" => Dict("E" => 30.0, "F" => 1.0, "V" => 1.0),
5                  "liquid" => Dict("E" =>  5.0, "F" => 0.5, "V" => 0.25) )
6  solver = BLR(tol = 1e-3, P = P)    # specify the solver, see Table I for options
7  acefit!(model, data, solver; weights=weights)    # solve lsq problem, update model parameters
8
9  # model accuracy on a test set
10 testdata = ... # load test data
11 errors(testdata, model)
12
13 # export the fitted potential
14 export2json("model.json", model)
15 export2lammps("model.yace", model)
```

Listing 8: Prototypical parameter estimation script, using some simple control over regression weights and solver parameters.

Next we discuss the minimization of the loss. Since all observations we consider here are linear, the minimization of $L(\boldsymbol{c})$ can be rewritten in the form

$$\arg\min_{\mathbf{c}} \left\| \mathbf{W}(\mathbf{y} - \mathbf{A}\boldsymbol{c}) \right\|^2, \tag{11}$$

where $\mathbf{y}$ is a vector containing the observation values $\mathcal{E}_R, \mathcal{F}_R, \mathcal{V}_R$, $\mathbf{A}$ is the design matrix containing the ACE basis values corresponding to those observations and $\mathbf{W}$ a diagonal matrix containing the weights $w_{E,R}, w_{F,R}, w_{V,R}$. Solving

`QR`	**QR decomposition:** Direct solution of the ridge regression problem (12). Tikhonov regularisation is imposed by extending the linear system. This method should rarely be used in practice and is included mostly for theoretical interest and the sake of completeness. `solver = QR(lambda = 0.0)`
`LSQR`	**Krylov method:** the standard iterative Krylov algorithm to solve the ridge regression problem (12). Tikhonov regularisation is imposed implicitly in the algorithm, with `damp` corresponding to the parameter λ. Early termination, by adjusting `atol` provides an additional and different form of regularisation. This algorithm is suitable for very large-scale parameter estimation problems. `solver = LSQR(damp = 1e-4, atol = 1e-6)`
`RRQR`	**Rank-revealing QR decomposition:** A random matrix sketching approach, which is computationally more efficient than the standard QR decomposition. In addition, the parameter `rtol` is closely related to λ in (12) but not identical. Instead of adding a Tikhonov term, RRQR regularisation is imposed by removing highly sensitive subspaces as determined by `rtol`. For large problems, this algorithm is more performant than the standard QR decomposition. `solver = RRQR(rtol = 1e-5)`
`BLR`	**Bayesian Linear Regression:** (or, Bayesian ridge regression) specifies a class of solvers that estimate regularisation hyperparameters, depending on the setting it estimates the scaling parameter λ or the entire Tikhonov matrix Γ. This solver also determines a posterior model distribution that can be used for uncertainty quantification. See Section II F for further details. This algorithm is more robust than `QR, LSQR, RRQR`, but computationally more intensive. It is highly recommended for relatively small datasets. `solver = BLR()`

TABLE I. Table of solvers for the ridge regression problem (12).

the linear least squares system (11) often results in overfitting, hence one almost always employs regularized methods, for example the ridge regression formulation,

$$\arg\min_{\mathbf{c}} \left\|\mathbf{W}(\mathbf{y} - \mathbf{A}\boldsymbol{c})\right\|^2 + \lambda\left\|\boldsymbol{\Gamma}\boldsymbol{c}\right\|^2, \tag{12}$$

where $\boldsymbol{\Gamma}$ specifies the form of the regularizer and λ a scaling parameter determining the relative weight of the regularisation. This formulation of the least squares problem is often also called regularized least squares, and the $\lambda\|\Gamma\boldsymbol{c}\|^2$ term is often called generalized Tikhonov regularisation. The default for $\boldsymbol{\Gamma}$ is zero or the identity, depending on the choice of solver. Our recommendation is to use the smoothness prior introduced in (10) instead for most solvers. Automatic relevance determination (ARD) is unique amongst the ridge regression solvers available in `ACEpotentials.jl` in that it estimates a regularizer $\boldsymbol{\Gamma}$ from the sensitivity of the parameters to the training data, at additional computational cost; see Section II F for more details.

To solve the ridge regression problem (12), `ACEpotentials.jl` employes the package `ACEfit.jl`[1], which offers a range of such algorithms. In the simplest setting, it can be used as shown in Listing 8, lines 6-7. For a list of the most important solvers, see Table I. For large models and/or large datasets, the parameter estimation task can be computationally challenging and may have to be performed on a cluster.

For small and moderate datasets we normally recommend the BLR method. For large datasets. when finely tuned regularisation is often less important, the random matrix sketching RRQR and iterative LSQR may be more appropriate.

Once the model parameters are determined as shown above, we typically wish to perform two tasks: (1) confirm the model accuracy on a test set; and (2) export the model to a format that can be used in standard MD codes, e.g., LAMMPS and ASE. Suppose that we are provided with a test data set `testdata`, then we can determine the model errors on that test set as seen in Listing 8, lines 9-11. This will print tables of RMSE and MAE errors for individual configuration types. If we wish to store and/or export the fitted potential for later use, we typically save it in `.json` format which can be read by `ACEpotentials.jl` as well as its Python interface to ASE, and in `.yace` format which can be read by the `pace` extension to LAMMPS; cf. Listing 8, lines 13-15.

[1] `https://github.com/ACEsuit/ACEfit.jl`

F. Bayesian framework for parameter estimation

Uncertainty estimates of model predictions are highly sought after tools to judge the accuracy of a prediction during simulation with a fitted model, but can also be employed to great effect during the model development workflow, e.g., in an active learning context. Such uncertainty estimates can be derived in a principled way by recasting the ridge regression problem (12) in a Bayesian framework where inference is based on the Bayesian posterior distribution

$$\mathrm{post}(\boldsymbol{c}) = p(\boldsymbol{c} \,|\, \mathbf{A}, \mathbf{y}) \propto p(\mathbf{A}, \mathbf{y} \,|\, \boldsymbol{c})\, p(\boldsymbol{c}). \tag{13}$$

Here, $p(\mathbf{A}, \mathbf{y} \,|\, \boldsymbol{c})$ denotes the likelihood of the observed data, and $p(\boldsymbol{c})$ the prior distribution on the model parameters. The Bayesian analogue of (12) is a Bayesian Linear Regression model with Gaussian observational noise and prior,

$$p(\mathbf{A}, \mathbf{y} \,|\, \boldsymbol{c}) \propto \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{A}\boldsymbol{c})^T(\beta\mathbf{W}^2)(\mathbf{y} - \mathbf{A}\boldsymbol{c})\right), \qquad \text{and} \tag{14}$$

$$p(\boldsymbol{c}) \propto \exp\left(-\frac{1}{2}\boldsymbol{c}^T\boldsymbol{\Sigma}_0^{-1}\boldsymbol{c}\right), \tag{15}$$

where the covariance $\beta^{-1}\mathbf{W}^{-2}$ of the observation noise depends on the regression weight matrix $\mathbf{W}$ and a hyper-parameter $\beta > 0$. This choice of prior and noise model yields a Gaussian posterior distribution, $p(\boldsymbol{c} \,|\, \mathbf{A}, \mathbf{y}) = \mathcal{N}(\boldsymbol{c}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, with mean and covariance given, respectively, by $\boldsymbol{\mu} = \beta\boldsymbol{\Sigma}\mathbf{A}^T\mathbf{W}^2\mathbf{y}$ and $\boldsymbol{\Sigma} = \left(\beta\mathbf{A}^T\mathbf{W}^2\mathbf{A} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1}$. We assume that the prior covariance $\boldsymbol{\Sigma}_0$ is of the form of a diagonal matrix. The above Bayesian model can be connected to the ridge regression formulation of equation (12) by noticing that maximising the posterior density (13) is equivalent to minimizing the regularized loss in (12) when $\boldsymbol{\Sigma}_0^{-1} = \zeta\boldsymbol{\Gamma}^2$ for some $\zeta > 0$ and $\lambda = \zeta/\beta$.

Solvers and model selection via evidence maximisation

The reliability of uncertainty estimates critically depends on the values of the model hyper-parameters, the noise and prior covariance matrices $\beta^{-1}\mathbf{W}^{-2}$ and $\boldsymbol{\Sigma}_0$. In ACE, it is sometimes difficult to make informed guesses of explicit values of these hyper-parameters that lead to good fits. We therefore commonly employ empirical Bayes approaches that infer appropriate values of these parameters directly from the training data by virtue of maximising the model evidence

$$\begin{aligned} p(\mathbf{A}, \mathbf{y} \,|\, \boldsymbol{\Sigma}_0, \beta) &= \int p(\mathbf{A}, \mathbf{y} \,|\, \boldsymbol{c}, \beta) p(\boldsymbol{c} \,|\, \boldsymbol{\Sigma}_0) d\boldsymbol{c} \\ &= \sqrt{\frac{\beta(2\pi)^{-N_{\mathrm{obs}}}|\boldsymbol{\Sigma}|}{|\boldsymbol{\Sigma}_0||\mathbf{W}^{-2}|}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{A}\boldsymbol{\mu})^T(\beta\mathbf{W}^2)(\mathbf{y} - \mathbf{A}\boldsymbol{\mu}) - \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}\right) \end{aligned} \tag{16}$$

as a function of $\boldsymbol{\Sigma}_0, \beta$. Intuitively, maximising the model evidence results in a model where the regularising effect of the covariance matrix $\boldsymbol{\Sigma}_0$ and the degree of penalisation of model misfit—modelled by the noise covariance matrix $\beta^{-1}\mathbf{W}^{-2}$—are balanced against the degree to which the regression coefficients are determined by the data.

Within `ACEpotentials.jl` this is implemented in the `BLR` solver (cf. Table I). Different solver options result in different constraints on the form of the prior covariance $\boldsymbol{\Sigma}_0$, and we refer to the documentation [22] for further details.

Uncertainty estimates via committees

Formally, the Bayesian ridge solver provides not an optimal parameter vector $\boldsymbol{c}$ but a posterior parameter distribution $p(\boldsymbol{c})$. In practice, one then selects the mean parameter vector $\boldsymbol{\mu}$ to specify the model. However, the posterior distribution remains important to estimate the uncertainty of predictions. Evaluating such uncertainties from the exact posterior distribution is computationally expensive; instead, `ACEpotentials` draws K samples $\{\boldsymbol{c}_k\}_{k=1}^K$ from $\mathrm{post}(\boldsymbol{c})$ resulting in a committee of ACE models which can be used to obtain computationally efficient uncertainty estimates for predictions. For example, the standard deviation σ of a total energy prediction can be approximated by a committee via

$$\tilde{\sigma}^2 = \frac{1}{\beta w_{E,R}^2} + \frac{1}{K}\sum_{k=1}^{K}(E^k - E^{\boldsymbol{\mu}})^2, \tag{17}$$

where $E^{\boldsymbol{\mu}}$ is the prediction made by the mean model with parameters $\boldsymbol{\mu}$, while E^k are the committee predictions from models with parameters $\boldsymbol{c}_k$. Similarly, uncertainty estimates can be made for any partial derivative of the potential energy surface such as for committee forces $F^k = \boldsymbol{c}_k \cdot \nabla \boldsymbol{B}_i$, or the mean force $F^{\boldsymbol{\mu}} = \boldsymbol{\mu} \cdot \nabla \boldsymbol{B}_i$.

The first term in (17) refers to the aleatoric, or irreducible, uncertainty arising due to randomness of the system which is dominated by the complexity of the linear ACE convergence parameters such as correlation order, polynomial degree and cutoff. The second term is the epistemic, or reducible, uncertainty arising due to a lack of data or rather information. An example how a variance estimate of the epistemic uncertainty can be obtained in the linear ACE framework is shown in Listing 9.

```
E, E_co = co_energy(model.potential, atoms)
sigma = sqrt( mean( (E_co .- E).^2 ) )
```

Listing 9: Example how to use a committee to estimate the uncertainty of a prediction. (Note that `model.potential` gives access to the calculator object.) Analogously, one can obtain committees of forces and virials.

III. WORKFLOW EXAMPLES

In this section, we present several practical examples of ACE usage, including simple benchmarks, practical potentials for materials and liquids to examples illustrating the hyperactive learning workflow. The scripts we used to generate the reported results are made available in a separate git repository[2] that will be regularly updated as the `ACEpotentials.jl` package evolves.

A. Tests with pre-existing data sets

1. *Benchmarks with limited-diversity datasets*

We test `ACEpotentials.jl` with default parameters on an early single-element benchmark dataset taken from [28]. This dataset was originally used to assess the relative strengths and weaknesses of four important MLIPs, the high-dimensional neural network potential (NNP)[29], the Gaussian approximation potential (GAP) [30], the Spectral Neighbor Analysis Potential (SNAP)[7], and moment tensor potentials (MTP)[8]. The benchmark contains six separate datasets corresponding to the six elements Li, Mo, Ni, Cu, Si and Ge, spanning a variety of chemistries (main group metal, transition metal and semiconductor), crystal structures (bcc, fcc, and diamond) and bonding types (metallic and covalent). For each element, the dataset contains the ground-state crystal structure, strained structures with strains of -10% to 10%, slab structures up to a maximum Miller index of three, and NVT ab initio molecular dynamics simulations of the bulk supercells with and without a single vacancy. These datasets contain a relatively large number of training structures, but only limited diversity.

In table II we see the comparison of the MAEs in energies and forces for the best performing potentials in the benchmark (GAP and MTP) with two linear ACE models trained with the default parameters and total degrees chosen to reach basis sizes of, respectively, 300 basis functions for **ACE(s)** and approximately 1000 basis functions for **ACE(l)**. We optimized none of the hyperparameters and solved used RRQR to estimate the parameters. We chose RRQR since the datasets are very large, hence a highly tuned regularisation is less important. This results in competitive accuracy across the entire benchmark. The only small exception is the slightly larger energy error for Mo-ACE(l), which suggests some fine-tuning of the model parameters could be beneficial in this particular case. Our aim with this experiment was to demonstrate that, with only minimal effort, linear ACE models can perform with (near-) best accuracy in a data set geared towards testing statistical generalization.

2. *Silicon*

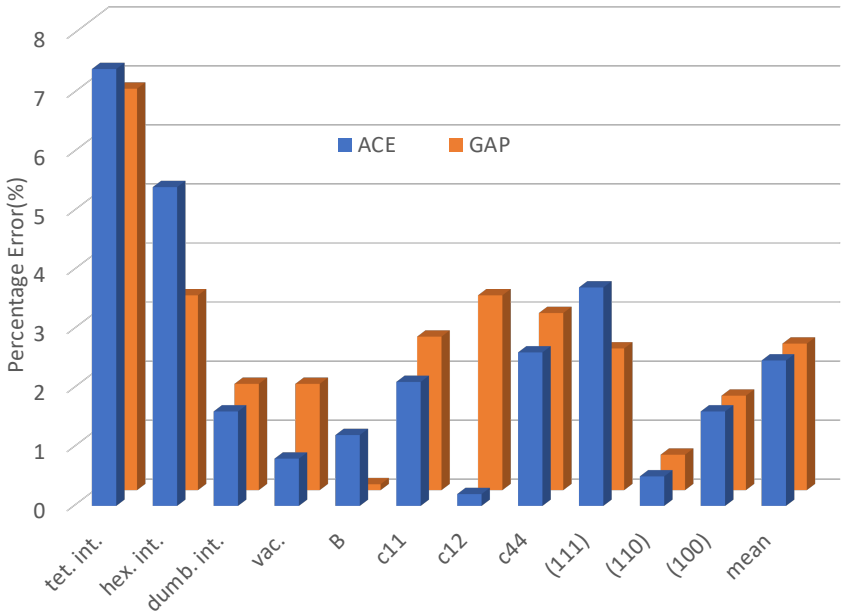
We used `ACEpotentials.jl` to fit a linear ACE potential to the silicon dataset introduced by Bartók et al [26] for fitting a Gaussian approximation potential (GAP). This extensive database contains a wide range of configurations

[2] `https://github.com/ACEsuit/ACEworkflows`

	Energy [meV]					Forces [eV/Å]			
	ACE(sm)	**ACE(lge)**	**GAP**	**MTP**		**ACE(sm)**	**ACE(lge)**	**GAP**	**MTP**
Ni	0.416	0.34	0.42	0.48	Ni	0.018	0.015	0.02	0.01
Cu	0.292	0.228	0.46	0.41	Cu	0.007	0.005	0.01	0.01
Li	0.231	0.165	0.49	0.49	Li	0.006	0.005	0.01	0.01
Mo	2.597	2.911	2.24	2.83	Mo	0.123	0.097	0.09	0.09
Si	3.501	1.985	2.91	2.21	Si	0.086	0.066	0.07	0.06
Ge	2.594	2.162	2.06	1.79	Ge	0.064	0.051	0.05	0.05

TABLE II. Mean absolute test errors in predicted energies and forces of two ACE models, ACE(sm) with ca 300 basis functions and ACE(lge) with ca 1000 basis functions, compared against the two best performing MLIPs published in [28].

ranging from several bulk crystal structures (diamond, hcp, fcc, etc.), amorphous structures as well as liquid MD snapshots, aiming to cover as much of the silicon energy landscape as possible. The corresponding GAP model was shown to outperform a wide range of other (classical) interatomic potentials on a large selection of accuracy and property or generalisation tests ranging from surface formation energies as well as liquid and radial distribution functions. The current work benchmarks an `ACEpotentials.jl` model, with default model parameters, containing basis functions up to order $\bar{\nu} = 4$, polynomial total degree $D^{\max} = 20$ and 6 Å cutoff against this silicon GAP potential. The model was fitted using generalised Tikhonov regularisation (12) of $\lambda\Gamma$, where Γ was constructed using an algebraic smoothness prior (10) with $p = 5$, whilst the `BLR` solver was used to estimate the scaling parameter λ. This benchmark is formed of a series of property tests including bulk diamond elastic constants, vacancy formation energies, surface formation energies for the (100), (110), (111) surfaces and hexagonal, dumbbell and tetragonal point defect energies for bulk diamond. These results of these property tests for the CASTEP [31] DFT reference, GAP and ACE are shown in Figure 2 and indicate good accuracy across the range of property tests. Percentage errors relative to the DFT reference are also included, confirming similarly accurate performance between the GAP and the `ACEpotentials.jl` frameworks.



	Point defects [eV]				Elastic Properties [GPa]				Surface energy [J/m^2]		
	tet. int.	hex. int.	dumb. int.	vac.	B	c_{11}	c_{12}	c_{44}	(111)	(110)	(100)
DFT	3.91	3.72	3.66	3.67	88.6	153.3	56.3	72.2	1.57	1.52	2.17
					Relative Error [%]						
GAP	6.8	3.3	1.8	1.8	0.1	2.6	3.3	3	2.4	0.6	1.6
ACE	7.4	5.4	1.6	0.8	1.2	2.1	0.2	2.6	3.7	0.5	1.6

FIG. 2. Benchmark of the silicon GAP [26] and ACE model presented in this work. Percentage relative errors against the DFT reference are provided in the Table

We also used this silicon ACE potential to carry out a more challenging test, namely to simulate fracture in the $(111)[1\bar{1}0]$ cleavage system. We used the `matscipy` package to setup a $12 \times 11 \times 1$ supercell containing 1586 atoms and to carry out structural optimisations with a Mode I crack anisotropic continuum linear elastic displacement field [32] applied with stress intensity factors ranging from $0.6K_G$ to $1.5K_G$ (where K_G is the Griffith load at which fracture becomes thermodynamically favourable). We observed spontaneous formation of the Pandey 2×1 reconstructed (111) surface behind the crack tip, in good agreement with previous studies using DFT [33] and GAP [26]. The critical stress intensity factor was determined to be $K_I = 1.0 \pm 0.02K_G$, which is very close to the expected Griffith value, indicating minimal lattice trapping. Overestimating the extent of lattice trapping is a common failure mode of previous interatomic potentials when applied to model fracture [34]. The total simulation time was around 30 minutes on a 28-core workstation.

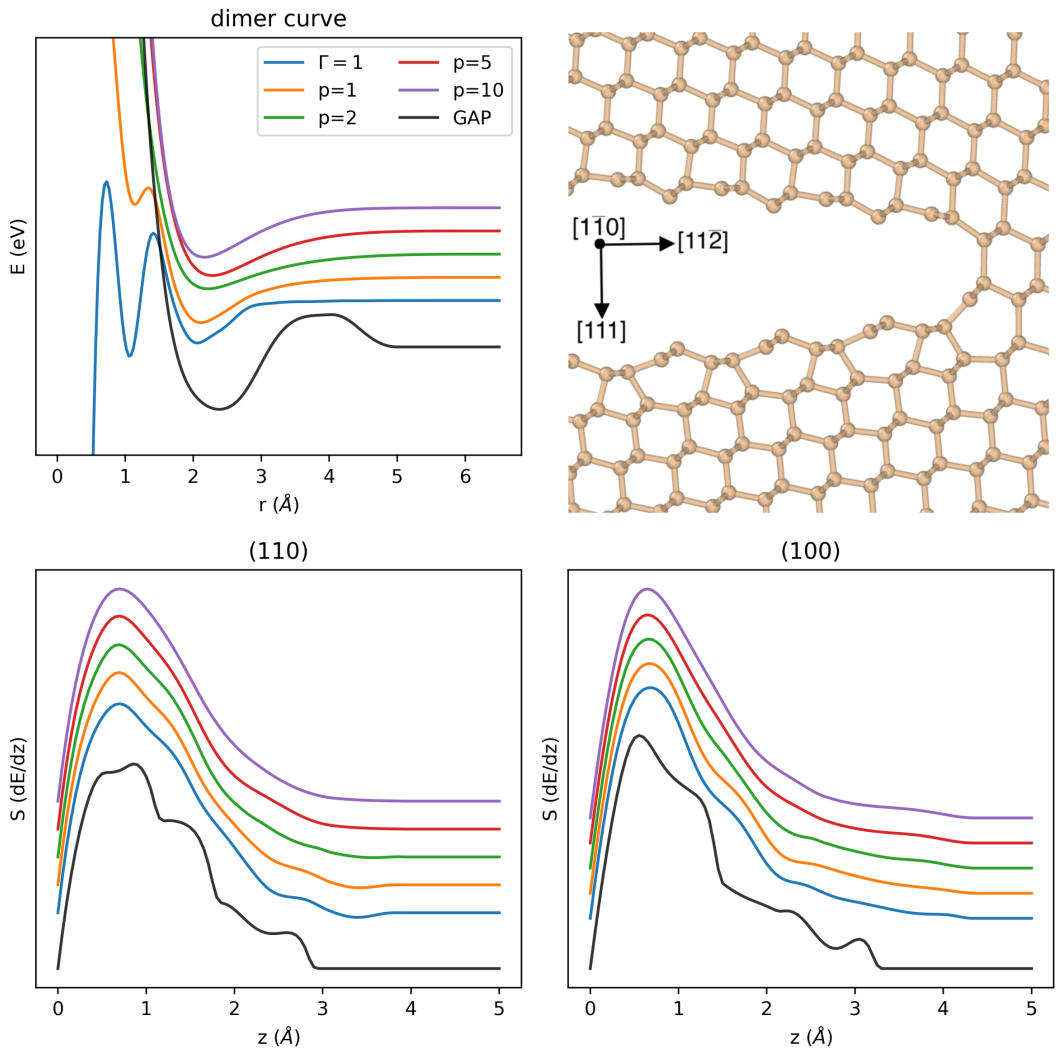


FIG. 3. **Top Left**: The predicted energy of the Si-Si dimer is shown for a sequence of ACE potentials trained with varying strengths of smoothness prior but equal accuracy (Force RMSE $\approx$ 0.075 eV/Å). $\Gamma = 1$ corresponds to an equal prior for all basis functions whilst p indicates the strength of the algebraic smoothness prior defined in (10). The black curve shows the corresponding result using GAP. All curves are shifted for clarity. **Bottom**: The evolution of stress (S) as a function of separation (z) during rigid decohesion of bulk silicon into the unrelaxed (110) and (100) surfaces is shown for the same sequence of potentials. **Top Right**: Snapshot from $\mathrm{Si}(111)[1\bar{1}0]$ quasi-static fracture simulation at a stress intensity factor of $1.8K_G$ using our ACE potential. The lower fracture surface shows a 2×1 Pandey reconstruction (alternating pentagons and heptagons), consistent with previous studies using DFT and GAP models, but at much reduced cost. The critical fracture toughness is very close to K_G, showing minimal lattice trapping.

To successfully carry out the fracture test it was crucial to produce a highly regular (smooth) ACE potential.

To illustrate the effect of changing the smoothness prior, a sequence of ACE potentials (order $\bar{\nu} = 4$, total degree $D^{\max} = 21$ and 6 Å cutoff), was fitted using no smoothness prior ($\Gamma = 1$) and increasing strengths of algebraic smoothness prior (10), $p = 1, 2, 5$ and 10. In all cases the model parameters were estimated using generalized Tychonov regularisation (12) with the scale factor λ tuned such that all potentials achieved a force RMSE of approximately 0.075 eV/Å , which is approximately 5% larger than without any regularisation. The effect of the prior on predicted Si-Si dimer curves and rigid bulk Si decohesion curves, which respectively probe smoothness of 2-body and many-body terms, is shown in Figure 3. Applying a moderate smoothness prior aids extrapolation into the close-approach region and reduces the amplitude of spurious oscillations seen in the stress (S) during decohesion.

3. Water

We investigated the ability of `ACEpotentials.jl` to capture the interactions in complex molecular liquids and to perform robust molecular dynamics simulations in such systems, fitting a linear ACE potential to a dataset containing 1593 liquid water configurations [35]. We chose only default model parameters, containing basis functions up to correlation order $\bar{\nu} = 3$, polynomial total degree $D^{\max} = 15$ and $r_{\text{cut}} = 5.5$ Å cutoff. Parameter estimation was performed using ARD with relevance threshold set by minmising the Bayesian Information Criterion (BIC) [36]. The training RMSE were 1.732 meV/atom for energies and 0.099 eV/Å for forces. To investigate the performance and robustness of the fitted ACE model, a series of mean squared displacement (MSD) simulation were performed under 1 bar NPT conditions at 300 K. The simulations were performed using 5184 atom simulation boxes, shown in Fig. 4 below, with PACE-LAMMPs [12]. The total simulation time for each of these simulation was 20 minutes utilising 1280 cores on ARCHER2, illustrating the efficiency of ACE potentials. The diffusion constant predicted by this simulation was 1.20 ± 0.03 m/s^2. It should be noted that diffusion constants are notoriously difficult to accurately determine especially considering the absence of long-range interactions into these ACE models. This example is therefore mostly an illustration of robustness and performance.

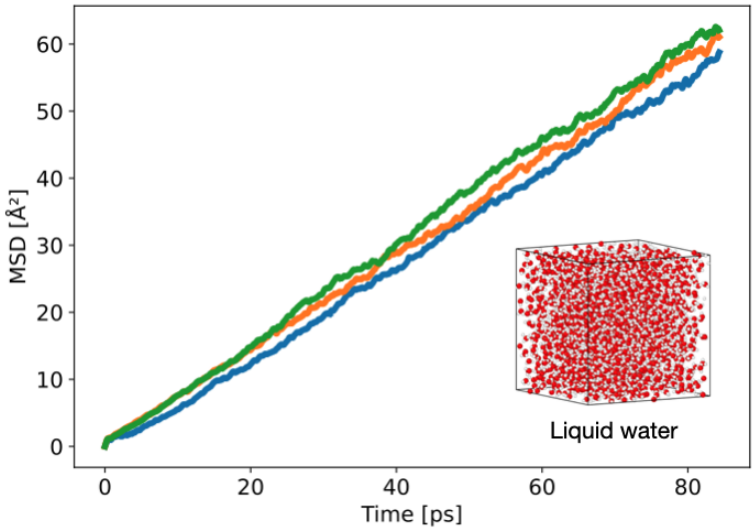


FIG. 4. Mean squared displacement (MSD) for three liquid water simulation at 1 bar NPT simulations and 300 K. The simulation cell contained 5184 atoms.

B. The Hyperactive Learning (HAL) Workflow

While fitting ACE potentials to pre-existing or "manually" assembled datasets, as discussed in Section III A, the real benefit of the linear ACE framework is in the construction robust and computationally inexpensive ACE potentials from the ground up with automated dataset assembly. This is achieved through the use of an iterative loop employing an active learning (AL) type approach [37, 38], where relevant training configurations are sampled to form a training database. To accelerate this AL process, hyperactive learning (HAL) [14] we introduced, which adds a biasing term to a molecular dynamics simulation towards predicted high uncertainty σ, as shown in (18). A tunable parameter

τ controls the strength of the biasing and thus the balance between physical exploration (molecular dynamics) and discovery of new structures (biasing).

$$E^{\mathrm{HAL}} = E^{\mathrm{ACE}} - \tau\sigma. \tag{18}$$

The HAL framework shares similarities with Bayesian Optimization (BO) as the biasing term is formally equivalent to a Lower Confidence Bound (LCB) acquisition function [39]. Similarly to BO, the parameter τ adjusts the tradeoff between exploration and exploitation during the generation of training configurations using HAL. HAL-generated configurations are both energetically reasonable, guided by E^{ACE} (exploitation), and informative, predicted by a relatively large value of σ (exploration). The bias towards uncertainty, mediated by an emerging biasing force during HAL dynamics, can be viewed as a strategy to acquire information (gain) by seeking out unseen (local) environments. The HAL approach can also be viewed as an adversarial attack, aimed to destabilize a fitted ACE potential such that, after iteratively adding sufficiently many new configurations, the linear ACE model is robust to such attacks which all but guarantees stable dynamics over long timescales.

The biasing parameter τ in HAL necessitates careful tuning, which HAL achieves through an adaptive scheme [14] that tunes τ on the fly by balancing the magnitude of the biasing force relative to the forces obtained by E^{ACE}. The *relative biasing parameter* τ_r used in this scheme is typically set to 0.1 to 0.2 and ensures that the biasing strength is reduced or increased depending on the degree of predicted uncertainty explored during the dynamics.

To initiate HAL, an initial database is typically constructed consisting of 1-10 configurations that sketch out some aspects of the energy landscape that are of interest to the application at hand. An ACE potential is fitted using a variant of the `BLR` solver, after which committee parameterisations $\{\mathbf{c}_k\}_{k=1}^{K}$, typically $K = 8$, are sampled from the posterior as discussed in Section II F. Biased MD/MC dynamics are then performed on E^{HAL}, using the dynamically tuned τ parameter. During the dynamics the relative force uncertainty f_i is recorded and once it exceeds a predefined tolerance f^{tol} a DFT calculation is triggered, and the training database is extended. This relative force uncertainty f_i is defined as

$$f_i = \frac{\frac{1}{K}\sum_{k=1}^{K} \|F_i^k - F_i^{\boldsymbol{\mu}}\|}{\|F_i^{\boldsymbol{\mu}}\| + \varepsilon}, \tag{19}$$

where F_i^k are the forces as obtained by the committee and $F_i^{\boldsymbol{\mu}}$ the forces predicted by the mean $\boldsymbol{\mu}$ of the posterior over the coefficients as outlined in Sec. II F. ε is a regularising constant used to regularize the fraction typically set to 0.2-0.4 eV/Å. Careful tuning of f_{tol} is required as it tunes the degree of extrapolation when adding new (unseen) configurations to the training database. Too large f_{tol} may lead to the sampling of energetically unreasonable configurations, whereas too small f_{tol} leads to suboptimal information gain during the HAL scheme resulting in sampling unnecessarily many configurations. The HAL scheme is outlined in Figure 5 illustrating how from a small initial training database containing a handful of configurations of interest a stable ACE potential is generated by performing biased MD and MC steps and iteratively triggering DFT calculations. For future reference, we define a *HAL iteration* to consist of (i) a biased MD simulation run until a new unseen structure is flagged, (ii) evaluating energies, forces and virials on the new structure, and (iii) updating the ACE potential model.

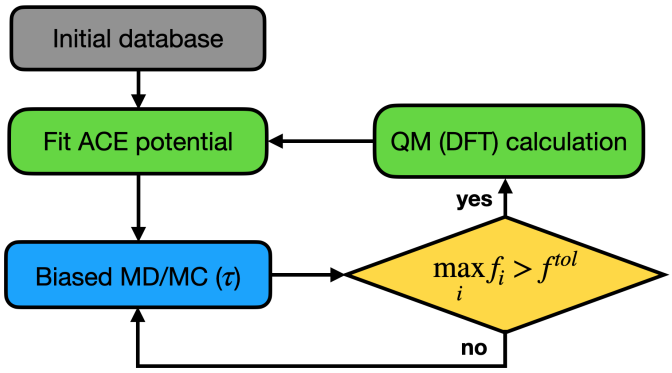


FIG. 5. Hyperactive Learning (HAL) protocol. Linear ACE potentials are fitted using BRR or ARD after which biased MD/MC steps are performed controlled by biasing parameter τ. Once the uncertainty metric f_i exceeds f^{tol} a DFT calculation is triggered a HAL iteration is completed and the training database extended.

1. AlSi10 melting temperature

The HAL framework was used to create an ACE potential for determining the melting temperature of the AlSi10 alloy. An initial dataset consisted of 32-atom random fcc lattice configurations, each containing 98 aluminium and 10 silicon atoms. This initial dataset was composed of 5 fcc random alloy configurations with lattice constants ranging from 14.3 to 16.6 Å^3/atom. The ACE basis set included interactions up to correlation order $\bar{\nu} = 2$ (3-body), and employed a cutoff of 5.5 Å. The model was fitted using Automatic Relevance Determination (ARD) and its sparsity set by minimising BIC which resulted in increasingly complex ACE models as more configurations (or information) were . The chosen maximum polynomial degree $D^{\max}$ during the HAL procedure increased from 4 to 12. The parameter estimation was carried out using ARD. The HAL relative biasing strength was set to $\tau_r = 0.2$, and the relative uncertainty threshold to $f^{\text{tol}} = 0.2$.

The HAL dynamics was used to melt the random alloy crystal structure, by ramping the temperature from 0 K to 1500 K at 1 GPa using a 1 fs timestep. Cell swapping and volume adjusting HAL-MC steps were taken to facilitate exploration of the (biased) energy landscape. After 18 HAL iterations, the ACE potential was already able to consistently perform 5000 HAL MD/MC timesteps without encountering new structures with high uncertainty. This final ACE potential contained 79 basis functions as selected using ARD pruning.

During these 18 HAL iterations the dimer curves are typically examined to ensure the potentials exhibit attraction at typical interatomic distances and short range repulsion as illustrated in Fig. 6.

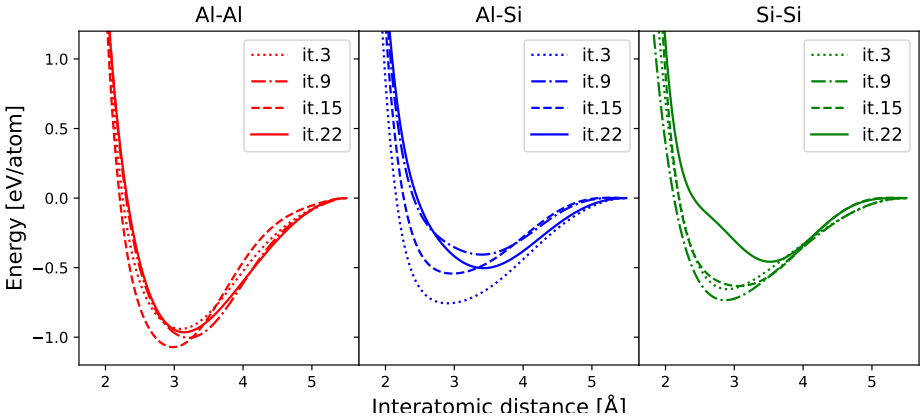


FIG. 6. ACE dimer curves for pair interactions for several HAL iterations. Stronger colours indicate later HAL iterations. They key observation to be drawn from this figure is that even in the early stages of the HAL process with very little available data, our priors ensure that the dimer curves are physically sensible, in particular smooth and repulsive.

The ACE potential obtained after HAL iteration 18 (fitted to 22 structures in total) was subsequently used to perform nested sampling (NS) simulations to model the liquid-solid phase transition. NS simulations were performed using 384 NS walkers, using a total decorrelation length of 512 formed by volume/shear/stretch/swap MC steps at a ratio of 4:4:4:4. The resulting heat capacity curves obtained by NS are presented in Figure 7 and are in close agreement to the melting temperature of 867 K as given by Thermo-Calc using the TCAL4 database [40].

2. Polyethylene glycol

The HAL framework [41] was used to create a polyethylene glycol (PEG) model. To initilize HAL, 18 structures of PEG(n=32) formed of 32 monomer units in vacuum were evaluated using the ORCA code [42] with the ωB97X DFT exchange correlation functional [43] and the 6-31G(d) basis set. ACE models were fitted to the initial and subsequent datasets with correlation order $\bar{\nu} = 3$, total degree $D^{\max} = 12$ and a cutoff radius 5.5 Å, using the ARD algorithm. The HAL protocol used relative biasing parameter $\tau_r = 0.1$ and uncertainty tolerance $f^{\text{tol}} = 0.3$ and performed at 500 K. Unlike the previous AlSi10 example, no cell adjusting or atom swapping HAL-MC steps were performed as the configurations are isolated molecules in vacuum. It was also chosen to not change the ACE basis throughout the HAL procedure but rather to keep it constant (e.g. $D^{\max} = 12$) as the initial database was relatively diverse. After 50 HAL iterations an ACE potential was generated that was deemed stable as it completed 10^4 HAL biased MD steps without triggering a DFT calculation. It was then used to determine the density of a PEG polymer formed of $n = 200$ monomer units in LAMMPS under periodic boundary conditions using the PACE evaluator [12]. The PEG(n=200)

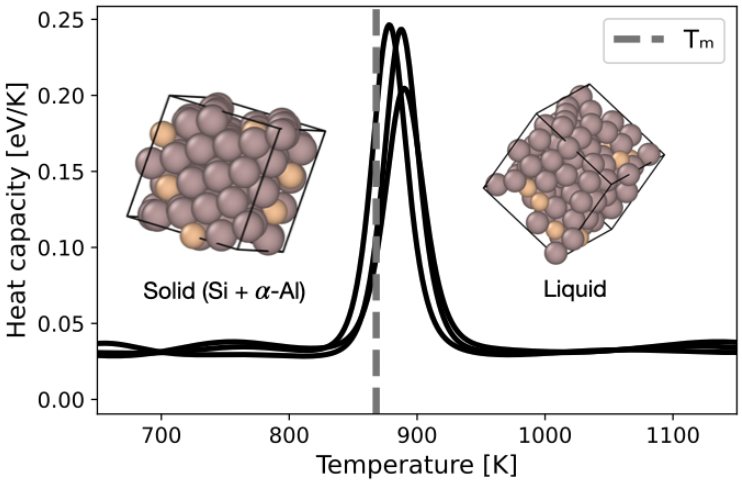


FIG. 7. NS AlSi10 heat capacity curves for several runs indicating the liquid-solid transition as predicted by the HAL generated ACE potential.

density was determined at 300 K, 350 K and 400 K at 1 bar pressure over a timescale of 0.5 ns as shown in Figure 8. The density at 300 K is in good agreement with the experimental density of 1.2 g/cm^3 [44] at 293 K. This illustrates remarkable extrapolative performance by the linear ACE framework as the DFT reference (ORCA) does not support periodic boundary conditions itself, making determining the PEG density purely from first principles impossible.

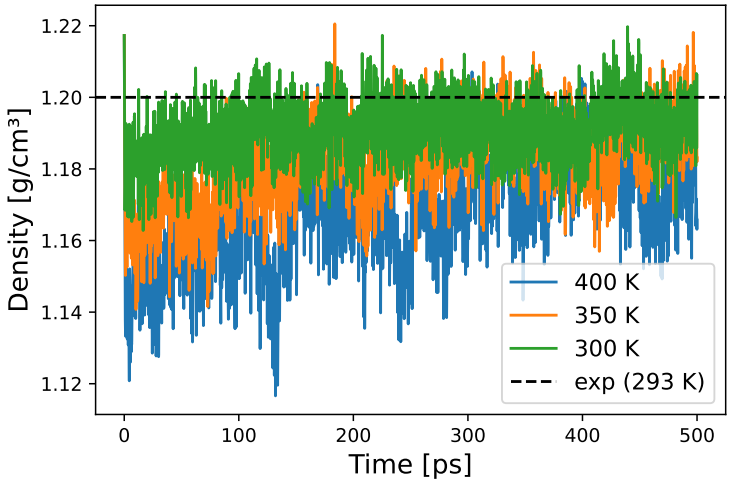


FIG. 8. PEG(n=200) density for HAL generated ACE potential under periodic boundary conditions using LAMMPs.

3. *Perovskite $CsPbBr_3$*

We used the HAL framework [41] to create a training dataset for the lead-halide perovskite $CsPbBr_3$, which shows three relevant phases: orthorhombic at low temperatures, tetragonal at intermediate temperatures, and cubic at high temperatures, with experimental transition temperatures of 361 K and 403 K [45]. The HAL process was designed to sample all of these phases so that the resulting potential accurately represents energy and entropy of each phase and is hence capable of predicting the transition temperatures. To ensure consistent DFT energies and effective vibrational mode sampling, approximately cubic 40 atom supercells were created for all three phases.

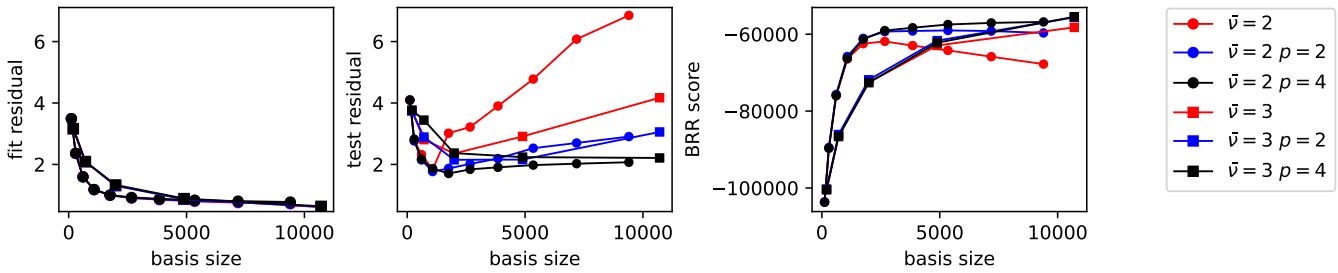


FIG. 9. Fitting set residual (left), testing set residual (center), and log marginal likelihood (right) as a function of basis size for $CsPbBr_3$ ACE model fit to a database generated with HAL. Symbol indicates correlation order $\bar{\nu}$, and color indicates smoothness prior exponent p.

This problem required some refinement of the standard HAL procedure, and careful testing of fitted ACE potentials for several basis sizes. We therefore give more detail about the process than in the previous cases.

The initial fit starting the HAL process used a set of 15 randomly perturbed (unit cell and atomic positions) 40-atom configurations, three from each of the high symmetry phases. The default ACE basis was used, with a cutoff of 8 Å, a smoothness prior with $p = 3$, and the `sklearn BayesianRidge` linear solver. Automated basis selection was applied every 10 HAL iterations, with a maximum basis size of 2000, $\bar{\nu} = 3$, a maximum total polynomial degree of 16, and the model score as the selection criterion. To encourage exploration of a wide range of temperatures and configurations, over a maximum of 10^4 1 fs HAL MD steps the temperature was ramped from 200 K to 600 K, and τ_r from 0.1 to 0.5. New fitting configurations were selected when the fractional force error f^{tol} exceeded 0.4. After 20 iterations starting from the three unperturbed high-symmetry 40-atom cells at fixed unit cell shape and size, the process was restarted from 9 80-atom high symmetry cells, doubling each of the three 40-atom cells along each cell vector, for 20 additional iterations. Then 20 additional iterations were carried out with variable unit cell and an applied pressure of 0.

At this point the model appeared to be stable enough for 10^5 steps without a HAL bias, so we switched to an unbiased sampling process to gather more data and improve the model accuracy. Starting the fit from the complete set of configurations from the HAL process, we generated fitting configurations from 2000 step runs with a maximum basis size of 4000. These used the same 80 atom starting configurations, but at fixed temperatures of 200 K to 500 K at 100 K intervals, and fixed shape but variable unit cell volume. To further refine the performance of the low energy parts of the PES around each high symmetry structure, we sampled 36 more configurations, each with 160 atoms (the three 40 atom supercells doubled along each of the three pairs of lattice vectors) at a range of lower temperatures, 150 K to 300 K at 50 K intervals.

The original set of 15 randomly perturbed configurations, another similar set of 15, and the 168 HAL configurations were used as the reference database for a set of fits to explore the performance of the model for a wide range of basis sizes. At this stage we filtered out physically unreasonable fitting data, as defined by a criterion that excluded any force larger than 10 eV/Å, as well as the energies and virials from such configurations. To fit the model and evaluate its predictive accuracy we split the set of configurations into 75% fitting and 25% testing, stratifying the split by the HAL iteration (or initial random perturbation set) that produced the configuration. The same fitting procedure and basis as in the HAL run were used, with $\bar{\nu} = 2$ and $\bar{\nu} = 3$ and maximum polynomial degree 4 to 16, up to a maximum basis size of 2×10^4. We also compared three choices for the smoothness prior: none, $p = 2$, and $p = 4$.

The training set residuals, test set residuals, and `BayesianRidge` score (log marginal likelihood) are plotted as a function of basis size in Fig. 9. For each value of $\bar{\nu}$ the fitting error improves monotonically as the basis size (and polynomial degree) increases, but at equal basis size the $\bar{\nu} = 2$ residuals are lower by as much as 25% (especially for moderately sized bases), indicating that for this system increasing the polynomial degree provides the basis with more useful flexibility as compared to increasing $\bar{\nu}$. For the basis size range where the error is minimized, the testing set residuals are larger than the fitting set by at least about a factor of 2, indicating that some amount of overfitting is occurring. The smoothness prior is successful at limiting the extent of this overfitting.

The generally lower training and test errors for the $\bar{\nu} = 2$ models relative to the correlation order three models are reflected in their Bayesian ridge scores (log marginal likelihoods). However, within each correlation order the optimal choice of polynomial degree and corresponding basis size indicated by the minimum test error are not consistent with the score. Indeed, the results displayed in Figure 9 lead us to conclude that the Bayesian ridge score is not always a reliable tool for optimal basis selection and other options should be explored in the future.

We used the model with lowest test set error, generated by the fit with $\bar{\nu} = 2$, maximum polynomial degree 12, and smoothness prior $p = 4$, to simulate larger unit cells of $CsPbBr_3$ at a range of temperatures spanning its expected range of phase transition temperatures. We simulated 32 independent constant temperature, constant pressure,

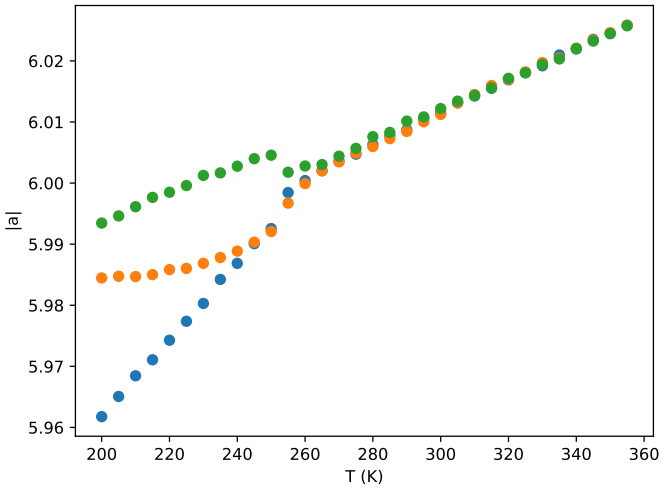


FIG. 10. Effective cubic lattice constants at fixed temperature simulated using the ACE model with $\bar{\nu} = 2$, maximum polynomial degree 12, and $p = 4$. All three values are identical (to within the estimated error) at $T > 255$ K indicating a cubic structure. At lower temperatures these split into a single value and a group of two, consistent with a tetragonal structure, and at $T < 240$ K they split further into three distinct values, consistent with an orthorhombic structure.

MD trajectories at temperatures from 200 K to 355 K and zero pressure for 10^4 10 fs time steps. Each trajectory started from an $8 \times 8 \times 6$ supercell (7680 atoms) of the orthorhombic structure. To analyze the resulting structure we reconstructed the effective cubic lattice vectors and averaged their magnitudes over the last 8000 steps of each trajectory. A plot of these effective cubic cell lattice vector magnitudes as a function of temperature is shown in Fig. 10. We see the three expected phases as indicated by the degeneracy of the lattice constants: cubic at high temperature, tetragonal at intermediate temperatures, and orthorhombic at low temperatures. The transition temperatures are 240 K and 255 K, which are substantially shifted relative to the experimental results of 361 K and 403 K [45]. We expect that this deviation from experiment is primarily due to our choice of exchange correlation functional, the Perdew-Burke-Enzerhof generalized-gradient approximation, [46] as has been seen in similar simulations [47]. A direct comparison to DFT would be useful, but it would require an accurate calculation of the predicted phase transition temperatures directly from the DFT PES, which is too computationally demanding to be practical without additional approximations.

IV. COMPUTATIONAL PERFORMANCE

The linearity of ACE potentials renders them not only interpretable but also efficient in terms of computational performance. To demonstrate this, a performance test was conducted on various linear ACE potentials referenced in this paper. The evaluation times, as well as some ACE hyperparameters used, are shown in Table III for the AlSi10, $CsPbI_3$, H_2O, PEG and Si potential developed in this work. The number of basis functions for each model is given too and may be fewer than a complete ACE basis parameterized by $\bar{\nu}$ and $D^{\max}$ due to ARD pruning basis functions with low relevance. The timings were obtained using the LAMMPs-PACE implementation [12] using a 128 core ARCHER2 node, equivalent to two seperate AMD EPYC 7742 64-core at 2.25GHz. The 10^6 steps/day figures are equivalent to a ns/day and were obtained for varying cell sizes to illustrate scaling. A standardized performance figure in the form of core-μs/atom figure is also provided. The silicon database fitted originates from the silicon GAP potential, whereas the AlSi10, PEG and $CsPbBr_3$ potentials were fitted using HAL generated databases containing 22, 68 and 198 configurations respectively as discussed in the previous subsections.

V. CONCLUSION AND OUTLOOK

We introduced `ACEpotentials.jl`, a front-end for several Julia-language packages that implement Atomic Cluster Expansion (ACE) MLIPs and related functionality. This front-end provides a user-oriented interface, while the backend packages combine excellent performance with the flexibility for rapid model development and experimentation that is typical for the Julia language. The front-end `ACEpotentials.jl` exposes a relatively simple subset of ACE type

	ACE parameters				Performance	
	$\bar{\nu}$	$D^{\max}$	r_{cut}	# basis func.	10^6 steps/day [atoms]	core-μs/atom
AlSi10	2	7	5.5	79	636 [32]	23
$CsPbBr_3$	2	12	5.5	544	334 [20]	93
PEG	3	12	5.5	4897	10 [1400]	227
Si	4	20	6	5434	7 [250]	744

TABLE III. Performance of linear ACE potentials for various systems using an ARCHER2 node utilising 128 cores for the 10^6 steps/day figures (equivalent ns/day using a 1 fs timestep). Core-μs/atom figures were obtained by performing simulations in serial.

models, linear models with robust priors, that we consider reliable in every-day use, especially in the context of an active learning type workflow.

However, we emphasize that the ACE framework allows for a much richer MLIPs design space [9, 12, 48–50] as well as parameterisation of many other types of particle systems [51–54]. We therefore conclude by mentioning some of those extensions, as well as current short-comings, that require further development.

- Robust parameter estimation, in particular hyperparameter tuning, remains under-investigated in the MLIPs context. We regularly experience that hand-tuned hyperparameters can give superior results, basis sparsification remains poorly understood, and uncertainties are often only indicative of actual errors. Further research is required to resolve these closely related issues.
- The design space of the `ACEpotentials.jl` ACE models can be expanded to admit trainable radial embeddings, composition of ACE features with nonlinearities, or even multi-layer architectures such as [48, 49]. This comes at the cost of highly nonlinear and less efficient models, but some of those extensions, such as trainable radial embeddings, can be undertaken while keeping the spirit of our current ACE models: small models for rapid iterative development and low evaluation cost.
- The extension to highly nonlinear models would likely require that the computational kernels on which `ACEpotentials.jl` is built also be made GPU-capable. Towards that end a deep learning framework such as MACE [49] (see also the `mace`[3] code) may be better suited.
- Finally, we note that there are already several related ACE software packages within `ACEsuit`[4] that implement a variety of models for other particle systems at different stages of development: Hamiltonians ([51], `ACEhamiltonians.jl`); wave functions ([53, 54], `ACEpsi.jl`); jet tagging models ([52], `BIPs.jl`). These build on an experimental and significantly expanded Julia-language ACE package `ACE.jl`.

ACKNOWLEDGMENTS

GC acknowledges support from EPSRC grant EP/X035956/1. CO, AR and TJ were supported by NSERC Discovery Grant GR019381 and NFRF Exploration Grant GR022937. WB was supported by US AFRL grant FA8655-21-1-7010. C vd O and GC acknowledge ARCHER2 for which access was obtained via the UKCP consortium and funded by EPSRC grant EP/P022065/1. NB was supported by the U. S. Office of Naval Research through the U. S. Naval Research Laboratory's fundamental research base program. EG acknowledges support from the EPSRC Centre for Doctoral Training in Automated Chemical Synthesis Enabled by Digital Molecular Technologies with grant reference EP/S024220/1. WCW was supported by the Schmidt Science Fellows in partnership with the Rhodes Trust, and additionally acknowledges support from EPSRC (Grant EP/V062654/1). JRK and CO acknowledge funding from the Leverhulme Trust under grant RPG-2017-191 and the EPSRC under grant EP/R043612/1. JRK, JPD and GC acknowledge support from the NOMAD Centre of Excellence funded by the European Commission under grant agreement 951786. JRK acknowledges support from the EPSRC under grants EP/P002188 and EP/R012474/1. This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2 funding from the EPSRC (capital grant EP/T022159/1), and DiRAC funding from the STFC (www.dirac.ac.uk). Further computing facilities were provided by the Scientific Computing Research Technology Platform of the University of Warwick.

[3] `https://github.com/ACEsuit/mace`
[4] `https://github.com/ACEsuit`

Appendix A: Linear Scaling Cost and Computational Kernels

In Sections II A and II B we outlined some basic ideas behind the ACE model, in particular expressing the potential energy model in terms of the many-body expansion (2). A naive implementation of the many-body expansion results in prohibitive computational cost due to the exponential cost of the sums over clusters $(j_1, \ldots, j_\nu)$. However, after discretizing the $U^{(\nu)}$-body potential of the *self-interacting many-body expansion* (2b) the sum can be rewritten to result in linear scaling cost. This is presented in detail, for example, in [9, 11, 12], hence we shall not review this process in full detail here. In order to outline what is involved in an implementation of an ACE potential, we only recall the form that the ACE model takes after this re-organisation of the many-body summation. The evaluation of the *self-interacting* ACE basis then results in the following stages:

1. Evaluation of the embeddings, $R_{nl}(r_{ij}, Z_i, Z_j)$ and $Y_l^m(\hat{\boldsymbol{r}}_{ij})$.

2. A pooling operation; also called called the atomic basis [9], or the density projection [2],

$$A^i_{znlm} = \sum_{j \in \mathcal{N}(i)} \phi_{znml}(\boldsymbol{r}_{ij}, Z_j, Z_i), \tag{A1}$$

   where $\mathcal{N}(i)$ denotes the set of indices of all atoms within the cutoff radius from atom i.

3. Product basis: for lexicographically ordered tuples $(\boldsymbol{z}, \boldsymbol{n}, \boldsymbol{l}, \boldsymbol{m}) = (z_t, n_t, l_t, m_t)_{t=1}^{\nu}$ we define

$$\boldsymbol{A}^i_{\boldsymbol{znlm}} = \prod_{t=1}^{\nu} A^i_{z_t n_t l_t m_t}. \tag{A2}$$

   This operation can be thought of as a sparse symmetric tensor product, or as taking ν-correlations.

4. Symmetrization: To ensure invariance one averages $\boldsymbol{A}^i$ over all rotations, resulting in the $O(3)$-invariant basis

$$\boldsymbol{B}^i = \mathcal{C}\boldsymbol{A}^i, \tag{A3}$$

   employed in the definition of the linear ACE model (3). Here, $\boldsymbol{A}^i$ is the vector of $(\boldsymbol{A}^i_{\boldsymbol{znlm}})$ basis functions while $\mathcal{C}$ a sparse matrix.

For each of these stages efficient computational kernels are implemented, designed in a modular way so that they can be independently optimized or composed into new model architectures.

Canonical Many-Body Expansion

Under the condition that the radial basis and envelope function are pure polynomials, it is possible to transform the self-interacting ACE basis $\boldsymbol{B}^i$ defined in (A3) into a basis for the canonical many-body expansion (2a). The idea behind this procedure is sketched out in [11]. The precise details of the implementation and a detailed study is not the purpose of this review. Here, we only mention that, upon slightly extending the R_{nl}, A^i and $\boldsymbol{A}^i$ bases, one can obtain a "purification operator" $\mathcal{P}$ such that the linearly transformed $\mathcal{P}\boldsymbol{A}^i$ becomes a basis for the canonical many-body expansion (2a). The symmetrisation $\mathcal{C}$ can then be applied to obtain an $O(3)$-invariant basis $\mathcal{B}^i := \mathcal{C}\mathcal{P}\boldsymbol{A}^i$.

An important variation of the "purification operation" $\mathcal{P}$ is to only purify the 2-body interaction. This entails replacing the fully self-interacting basis functions

$$\boldsymbol{A}^i_{\boldsymbol{k}} = \sum_{j_1,\ldots,j_\nu} \prod_{t=1}^{\nu} \phi_{k_t}(x_{ij_t}) \qquad \text{with} \qquad \sum_{\substack{j_1,\ldots,j_\nu \\ j_a \neq j_b}} \prod_{t=1}^{\nu} \phi_{k_t}(x_{ij_t})$$

All three options (i) fully self-interacting, (ii) purified pair interaction, and (iii) canonical cluster expansion are available in `ACEpotentials.jl`. The package documentation should be reviewed on how to select the different basis sets.

---

[1] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, Apr 2007.

[2] Albert P Bartók, Mike C Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13):136403, 2010.

[3] Volker L. Deringer, Miguel A. Caro, and Gábor Csányi. Machine learning interatomic potentials as emerging tools for materials science. *Advanced Materials*, 31(46):1902765, 2019.

[4] Jörg Behler and Gábor Csányi. Machine learning potentials for extended systems: a perspective. *The European Physical Journal B*, 94, 2021.

[5] Volker L. Deringer, Albert P. Bartók, Noam Bernstein, David M. Wilkins, Michele Ceriotti, and Gábor Csányi. Gaussian process regression for materials and molecules. *Chemical Reviews*, 121(16):10073–10141, 2021. PMID: 34398616.

[6] Felix Musil, Andrea Grisafi, Albert P. Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti. Physics-inspired structural representations for molecules and materials. *Chemical Reviews*, 121(16):9759–9815, 2021. PMID: 34310133.

[7] A.P. Thompson, L.P. Swiler, C.R. Trott, S.M. Foiles, and G.J. Tucker. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, 285:316–330, 2015.

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

[9] Ralf Drautz. Atomic cluster expansion for accurate and transferable interatomic potentials. *Phys. Rev. B*, 99:014104, Jan 2019.

[10] Atsuto Seko, Atsushi Togo, and Isao Tanaka. Group-theoretical high-order rotational invariants for structural representations: Application to linearized machine learning interatomic potential. *Physical Review B*, 99, 06 2019.

[11] Geneviève Dusson, Markus Bachmayr, Gábor Csányi, Ralf Drautz, Simon Etter, Cas van der Oord, and Christoph Ortner. Atomic cluster expansion: Completeness, efficiency and stability. *Journal of Computational Physics*, 454:110946, 2022.

[12] Yury Lysogorskiy, Cas van der Oord, Anton Bochkarev, Sarath Menon, Matteo Rinaldi, Thomas Hammerschmidt, Matous Mrovec, Aidan Thompson, Gábor Csányi, Christoph Ortner, and Ralf Drautz. Performant implementation of the atomic cluster expansion (pace) and application to copper and silicon. *npj Computational Materials*, 7(1):97, 2021.

[13] Dávid Péter Kovács, Cas van der Oord, Jiri Kucera, Alice E. A. Allen, Daniel J. Cole, Christoph Ortner, and Gábor Csányi. Linear atomic cluster expansion force fields for organic molecules: Beyond rmse. *Journal of Chemical Theory and Computation*, 17(12):7696–7711, 2021. PMID: 34735161.

[14] Cas van der Oord, Matthias Sachs, Dávid Péter Kovács, Christoph Ortner, and Gábor Csányi. Hyperactive learning (hal) for data-driven interatomic potentials, 2022.

[15] Y. Liang, M. Mrovec, Y. Lysogorskiy, M. Vega-Paredes, C. Scheu, and R. Drautz. Atomic cluster expansion for pt-rh catalysts: From ab initio to the simulation of nanoclusters in few steps. arXiv:2303.07465.

[16] Anton Bochkarev, Yury Lysogorskiy, Sarath Menon, Minaam Qamar, Matous Mrovec, and Ralf Drautz. Efficient parametrization of the atomic cluster expansion. *Phys. Rev. Mater.*, 6:013804, 2022.

[17] M. Qamar, M. Mrovec, Y. Lysogorskiy, A. Bochkarev, and R. Drautz. Atomic cluster expansion for quantum-accurate large-scale simulations of carbon. arXiv:2210.09161.

[18] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.*, 271:108171, 2022.

[19] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, Eric D Hermes, Paul C Jennings, Peter Bjerre Jensen, James Kermode, John R Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W Jacobsen. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017.

[20] `Molly.jl`: Molecular simulation in julia.

[21] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[22] `ACEpotentials.jl`. Documentation and user interface for Julia-language development of ACE potentials, `https://github.com/ACEsuit/ACEpotentials.jl`.

[23] James F. Ziegler, J. P. Biersack, and U. Littmark. *The Stopping and Range of Ions in Solids*. Pergamon.

[24] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, Eric D Hermes, Paul C Jennings, Peter Bjerre Jensen, James Kermode, John R Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann

Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W Jacobsen. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, jun 2017.

[25] Wojciech J Szlachta, Albert P Bartók, and Gábor Csányi. Accuracy and transferability of gaussian approximation potential models for tungsten. *Phys. Rev. B Condens. Matter*, 90(10):104108, September 2014.

[26] Albert P Bartók, James Kermode, Noam Bernstein, and Gábor Csányi. Machine learning a General-Purpose interatomic potential for silicon. *Phys. Rev. X*, 8(4):041048, December 2018.

[27] Albert P Bartók, Sandip De, Carl Poelking, Noam Bernstein, James R Kermode, Gábor Csányi, and Michele Ceriotti. Machine learning unifies the modeling of materials and molecules. *Sci Adv*, 3(12):e1701816, December 2017.

[28] Yunxing Zuo, Chi Chen, Xiangguo Li, Zhi Deng, Yiming Chen, Jörg Behler, Gá bor Csányi, Alexander V. Shapeev, Aidan P. Thompson, Mitchell A. Wood, and Shyue Ping Ong. Performance and cost assessment of machine learning interatomic potentials. *The Journal of Physical Chemistry A*, 124(4):731–745, jan 2020.

[29] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, Apr 2007.

[30] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical Review Letters*, 104(13), Apr 2010.

[31] Stewart J. Clark, Matthew D. Segall, Chris J. Pickard, Phil J. Hasnip, Matt I. J. Probert, Keith Refson, and Mike C. Payne. First principles methods using castep. *Zeitschrift für Kristallographie - Crystalline Materials*, 220(5-6):567–570, 2005.

[32] G C Sih, P C Paris, and G R Irwin. On cracks in rectilinearly anisotropic bodies. *Int. J. Fract. Mech.*, 1(3):189–203, September 1965.

[33] J R Kermode, T Albaret, Dov Sherman, Noam Bernstein, P Gumbsch, M C Payne, Gábor Csányi, and A De Vita. Low-speed fracture instabilities in a brittle crystal. *Nature*, 455(7217):1224–1227, October 2008.

[34] Erik Bitzek, James R Kermode, and Peter Gumbsch. Atomistic aspects of fracture. *Int. J. Fract.*, 191(1-2):13–30, February 2015.

[35] Bingqing Cheng, Edgar A Engel, Jörg Behler, Christoph Dellago, and Michele Ceriotti. Ab initio thermodynamics of liquid and solid water. *Proceedings of the National Academy of Sciences*, 116(4):1110–1115, 2019.

[36] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461 – 464, 1978.

[37] Jonathan Vandermause, Yu Xie, Jin Soo Lim, Cameron J. Owen, and Boris Kozinsky. Active learning of reactive bayesian force fields applied to heterogeneous catalysis dynamics of h/pt. *Nature Communications*, 13(1):5183, 2022.

[38] Evgeny V. Podryabinkin and Alexander V. Shapeev. Active learning of linearly parametrized interatomic potentials. *Computational Materials Science*, 140:171–180, 2017.

[39] Malthe K. Bisbo and Bjørk Hammer. Global optimization of atomic structure enhanced by machine learning. *Phys. Rev. B*, 105:245404, Jun 2022.

[40] M. Tang, P.C. Pistorius, and S. et al Narra. Rapid solidification: Selective laser melting of alsi10mg. *JOM*, 68, 2016.

[41] `ACEHAL`. Implementation in python, `https://github.com/libAtoms/ACEHAL`.

[42] Frank Neese. The orca program system. *WIREs Computational Molecular Science*, 2(1):73–78, 2012.

[43] Jeng-Da Chai and Martin Head-Gordon. Long-range corrected hybrid density functionals with damped atom–atom dispersion corrections. *Phys. Chem. Chem. Phys.*, 10, 2008.

[44] *Polyethylene glycol [MAK Value Documentation, 1998]*, pages 248–270. John Wiley and Sons, Ltd, 2012.

[45] Constantinos C. Stoumpos, Christos D. Malliakas, John A. Peters, Zhifu Liu, Maria Sebastian, Jino Im, Thomas C. Chasapis, Arief C. Wibowo, Duck Young Chung, Arthur J. Freeman, Bruce W. Wessels, and Mercouri G. Kanatzidis. Crystal growth of the perovskite semiconductor cspbbr3: A new material for high-energy radiation detection. *Crystal Growth & Design*, 13(7):2722–2727, 2013.

[46] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77:3865–3868, Oct 1996.

[47] Erik Fransson, Julia Wiktor, and Paul Erhart. Phase transitions in inorganic halide perovskites from machine learning potentials. *arXiv preprint arXiv:2301.03497*, 2023.

[48] A. Bochkarev, Y. Lysogorskiy, C. Ortner, G. Csanyi, and R. Drautz. Multilayer atomic cluster expansion for semi-local interactions. *Phys. Rev. Research*, 4, 2022.

[49] Ilyes Batatia, Dávid Péter Kovács, Gregor N. C. Simm, Christoph Ortner, and Gábor Csányi. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields, 2022.

[50] James P Darby, James R Kermode, and Gábor Csányi. Compressing local atomic neighbourhood descriptors. *npj Computational Materials*, 8(1):1–13, 2022.

[51] Liwei Zhang, Berk Onat, Genevieve Dusson, Gautam Anand, Reinhard J Maurer, Christoph Ortner, and James R Kermode. Equivariant analytical mapping of first principles hamiltonians to accurate and transferable materials models. *npj Computational Materials*, 8, 2022.

[52] J. M. Munoz, I. Batatia, and C. Ortner. Boost invariant polynomials for efficient jet tagging. *Mach. Learn.: Sci. Technol.*, 3, 2022.

[53] R. Drautz and C. Ortner. Atomic cluster expansion and wave function representations. arXiv:2206.11375.

[54] D. Zhou, H. Chen, C. Hin Ho, and C. Ortner. A multilevel method for many-electron schrödinger equations based on the atomic cluster expansion. arXiv:2304.04260.