# SANDIA REPORT

SAND2014-17862
Unlimited Release
Printed Sept. 2014

# Automated Algorithms for Quantum-Level Accuracy in Atomistic Simulations: LDRD Final Report

Aidan P. Thompson, Peter A. Schultz, Paul S. Crozier, Stan G. Moore, Laura P. Swiler, J. Adam Stephens, Stephen M. Foiles, Garritt J. Tucker

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.



Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2014-17862
Unlimited Release
Printed Sept. 2014

# Automated Algorithms for Quantum-Level Accuracy in Atomistic Simulations: LDRD Final Report

Aidan P. Thompson, Peter A. Schultz, Paul S. Crozier, Stan G. Moore
Multiscale Science Dept.
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1318

Laura P. Swiler, J. Adam Stephens
Optimization and Uncertainty Quantification
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1318

Christian R. Trott
Scalable Algorithms
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1318

Stephen M. Foiles
Computational Materials and Data Sciences
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1411

Garritt J. Tucker
Materials Science and Engineering
Materials Modeling for Extreme Environments
Drexel University
3141 Chestnut Street, LeBow 344
Philadelphia, PA 19104, USA
gtucker@coe.drexel.edu

## Abstract

This report summarizes the result of LDRD project 12-0395, titled "Automated Algorithms for Quantum-level Accuracy in Atomistic Simulations." During the course of this LDRD, we have developed an interatomic potential for solids and liquids called Spectral Neighbor Analysis Potential (SNAP). The SNAP potential has a very general form and uses machine-learning techniques to reproduce the energies, forces, and stress tensors of a large set of small configurations of atoms, which are obtained using high-accuracy quantum electronic structure (QM) calculations. The local environment of each atom is characterized by a set of bispectrum components of the local neighbor density projected on to a basis of hyperspherical harmonics in four dimensions. The SNAP coefficients are determined using weighted least-squares linear regression against the full QM training set. This allows the SNAP potential to be fit in a robust, automated manner to large QM data sets using many bispectrum components. The calculation of the bispectrum components and the SNAP potential are implemented in the LAMMPS parallel molecular dynamics code. Global optimization methods in the DAKOTA software package are used to seek out good choices of hyperparameters that define the overall structure of the SNAP potential. FitSnap.py, a Python-based software package interfacing to both LAMMPS and DAKOTA is used to formulate the linear regression problem, solve it, and analyze the accuracy of the resultant SNAP potential. We describe a SNAP potential for tantalum that accurately reproduces a variety of solid and liquid properties. Most significantly, in contrast to existing tantalum potentials, SNAP correctly predicts the Peierls barrier for screw dislocation motion. We also present results from SNAP potentials generated for indium phosphide (InP) and silica ($SiO_2$). We describe efficient algorithms for calculating SNAP forces and energies in molecular dynamics simulations using massively parallel computers and advanced processor architectures. Finally, we briefly describe the MSM method for efficient calculation of electrostatic interactions on massively parallel computers.

# Acknowledgments

This work was funded by the Computing and Information Sciences (CIS) Investment Area within the Laboratory Directed Research and Development (LDRD) Program at Sandia National Laboratories, Project Number 158794 and Title "Automated Algorithms for Achieving Quantum-level Accuracy in Atomistic Simulations." The authors thank the CIS committee and Program Manager Mary Gonzales for their support of this project. We thank all of our managers for their support of this activity. We thank Steve Plimpton for invaluable LAMMPS support.

# Contents

## Appendix

# List of Figures

# Chapter 1

# Introduction

Classical molecular dynamics simulation (MD) is a powerful approach for describing the mechanical, chemical, and thermodynamic behavior of solid and fluid materials in a rigorous manner [10]. The material is modeled as a large collection of point masses (atoms) whose motion is tracked by integrating the classical equations of motion to obtain the positions and velocities of the atoms at a large number of timesteps. The forces on the atoms are specified by an inter-atomic potential that defines the potential energy of the system as a function of the atom positions. Typical inter-atomic potentials are computationally inexpensive and capture the basic physics of electron-mediated atomic interactions of important classes of materials, such as molecular liquids and crystalline metals. Efficient MD codes running on commodity workstations are commonly used to simulate systems with $N = 10^5 - 10^6$ atoms, the scale at which many interesting physical and chemical phenomena emerge. Quantum molecular dynamics (QMD) is a much more computationally intensive method for solving a similar physics problem [21]. Instead of assuming a fixed interatomic potential, the forces on atoms are obtained by explicitly solving the quantum electronic structure of the valence electrons at each timestep. Because MD potentials are short-ranged, the computational complexity of MD generally scales as $O(N)$, whereas QMD calculations require global self-consistent convergence of the electronic structure, whose computational cost is $O(N_e^\alpha)$, where $\alpha = 2-3$ and $N_e$ is the number of electrons. For the same reasons, MD is amenable to spatial decomposition on parallel computers, while QMD calculations allow only limited parallelism.

As a result, while high accuracy QMD simulations have supplanted MD in the range $N = 10-100$ atoms, QMD is still intractable for $N > 1000$, even using the largest supercomputers. Conversely, typical MD potentials often exhibit behavior that is inconsistent with QMD simulations. This has led to great interest in the development of MD potentials that match the QMD results for small systems, but can still be scaled to the interesting regime $N = 10^5 - 10^6$ atoms.[5, 3, 20]. These quantum-accurate potentials require many more floating point operations per atom compared to simpler potentials, but they are still short-ranged. So the computational cost remains $O(N)$, but with a larger algorithm pre-factor. In this report, we present a new quantum-accurate potential called SNAP. We explain the mathematical structure of the potential and the way in which we fit the potential parameters to a database of quantum electronic structure calculations. We then present results for the SNAP potentials that we have developed for tantalum, indium phosphide, and silica. After these results, we briefly describe the implementation of the SNAP potential in the LAMMPS code, including serial and parallel performance and some other details of the SNAP algorithm. We conclude with a summary and future research directions.

# Chapter 2

# Mathematical Formulation

## 2.1 Bispectrum components

The quantum mechanical principle of near-sightedness tells us that the electron density at a point is only weakly affected by atoms that are not near. This provides support for the common assumption that the energy of a configuration of atoms is dominated by contributions from clusters of atoms that are near to each other. It is reasonable then to seek out descriptors of local structure and build energy models based on these descriptors. Typically, this is done by identifying geometrical structures, such as pair distances and bond angles, or chemical structures, such as bonds. Interatomic potentials based on these approaches often produce useful qualitative models for different types of materials, but it can be difficult or impossible to adjust these potentials to accurately reproduce known properties of specific materials. Recently, Bartok et al. have studied several infinite classes of descriptor that are related to the density of neighbors in the spherically symmetric space centered on one atom[5, 6, 4]. They demonstrated that by adding descriptors of successively higher order, it was possible to systematically reduce the mismatch between the potential and the target data. One of these descriptors, the bispectrum of the neighbor density mapped on to the 3-sphere, forms the basis for their Gaussian Approximation Potential (GAP)[5]. We also use the bispectrum, which we derive below, as the basis for our SNAP potential.

The density of neighbor atoms around a central atom $i$ at location $\mathbf{r}$ can be considered as a sum of $\delta$-functions located in a three-dimensional space:

$$\rho_i(\mathbf{r}) = \delta(\mathbf{r}) + \sum_{r_{ii'}<R_{ii'}} f_c(r_{ii'}) w_{i'} \delta(\mathbf{r} - \mathbf{r}_{ii'}) \tag{2.1}$$

where $\mathbf{r}_{ii'}$ is the vector joining the position of the central atom $i$ to neighbor atom $i'$. The $w_{i'}$ coefficients are dimensionless weights that are chosen to distinguish atoms of different types, while the central atom is arbitrarily assigned a unit weight. The sum is over all atoms $i'$ within some species-dependent cutoff distance:

$$R_{ii'} = \alpha_{cut}(r_i + r_{i'}) \tag{2.2}$$

The function $f_c(r)$ ensures that the contribution of each neighbor atom goes smoothly to zero at $R_{ii'}$, $r_i$ and $r_{i'}$ are species-dependent cutoff radii, and $\alpha_{cut}$ is a constant factor. The angular part of this

density function can be expanded in the familiar basis of spherical harmonic functions $Y_m^l(\theta,\phi)$. The radial component is often expanded in a separate set of radial basis functions that multiply the spherical harmonics. Bartok et al. made a different choice, mapping the radial distance $r$ within the cut-off distance on to a third polar angle $\theta_0$ defined by,

$$\theta_0 = \theta_0^{max} \frac{r}{R_{ii'}} \tag{2.3}$$

The additional angle $\theta_0$ allows the set of points $(\theta,\phi,r)$ in the 3D ball of possible neighbor positions to be mapped on to the set of points $(\theta,\phi,\theta_0)$ that are a subset of the 3-sphere. Points south of the latitude $\theta_0^{max}$ are excluded. It is advantageous to use most of the 3-sphere, while still excluding the region near the south pole where the configurational space becomes highly compressed.

The natural basis for functions on the 3-sphere is formed by the 4D hyperspherical harmonics $U^j_{m,m'}(\theta_0,\theta,\phi)$, defined for $j = 0, \frac{1}{2}, 1, \ldots$ and $m,m' = -j, -j+1, \ldots, j-1, j$ [37]. These functions also happen to be the elements of the unitary transformation matrices for spherical harmonics under rotation by angle $2\theta_0$ about the axis defined by $(\theta,\phi)$. When the rotation is parameterized in terms of the three Euler angles, these functions are better known as $D^j_{m,m'}(\alpha,\beta,\gamma)$, the Wigner $D$-functions, which form the representations of the $SO(3)$ rotational group [22, 37]. Dropping the atom index $i$, the neighbor density function can be expanded in the $U^j_{m,m'}$ functions

$$\rho(\mathbf{r}) = \sum_{j=0,\frac{1}{2},\ldots}^{\infty} \sum_{m=-j}^{j} \sum_{m'=-j}^{j} u^j_{m,m'} U^j_{m,m'}(\theta_0,\theta,\phi) \tag{2.4}$$

where each expansion coefficient $u^j_{m,m'}$ is given by the inner product of the neighbor density with the basis function. Because the neighbor density is a weighted sum of $\delta$-functions, each expansion coefficient can be written as a sum over discrete values of the corresponding basis function,

$$u^j_{m,m'} = U^j_{m,m'}(0,0,0) + \sum_{r_{ii'}<R_{cut}} f_c(r_{ii'}) w_{i'} U^j_{m,m'}(\theta_0,\theta,\phi) \tag{2.5}$$

The expansion coefficients $u^j_{m,m'}$ are complex-valued and they are not directly useful as descriptors, because they are not invariant under rotation of the polar coordinate frame. However, the following scalar triple products of expansion coefficients can be shown to be real-valued and invariant under rotation[4].

$$B_{j_1,j_2,j} = \sum_{m_1,m_1'=-j_1}^{j_1} \sum_{m_2,m_2'=-j_2}^{j_2} \sum_{m,m'=-j}^{j} (u^j_{m,m'})^* H^{jmm'}_{j_1 m_1 m_1' \; j_2 m_2 m_2'} \; u^{j_1}_{m_1,m_1'} u^{j_2}_{m_2,m_2'} \tag{2.6}$$

The constants $H^{jmm'}_{j_1m_1m'_1\ j_2m_2m'_2}$ are coupling coefficients, analogous to the Clebsch-Gordan coefficients for rotations on the 2-sphere. These invariants are the components of the bispectrum. They characterize the strength of density correlations at three points on the 3-sphere. The lowest-order components describe the coarsest features of the density function, while higher-order components reflect finer detail. An analogous bispectrum can be defined on the 2-sphere in terms of the spherical harmonics. In this case, the components of the bispectrum are a superset of the second and third order bond-orientational order parameters developed by Steinhardt et al.[32]. These in turn are specific instances of the order parameters introduced in Landau's theory of phase transitions[18].

The coupling coefficients are non-zero only for non-negative integer and half-integer values of $j_1, j_2$, and $j$ satisfying the conditions $\|j_1 - j_2\| \leq j \leq j_1 + j_2$ and $j_1 + j_2 - j$ not half-integer[22]. In addition, $B_{j_1,j_2,j}$ is symmetric in $j_1$ and $j_2$. Hence the number of distinct non-zero bispectrum components with indices $j_1, j_2, j$ not exceeding a positive integer $J$ is $(J+1)^3$. Furthermore, it can be shown [35] that bispectrum components with reordered indices are related by the following identity:

$$\frac{B_{j_1,j_2,j}}{2j+1} = \frac{B_{j,j_2,j_1}}{2j_1+1} = \frac{B_{j_1,j,j_2}}{2j_2+1}. \tag{2.7}$$

We can exploit this equivalence by further restricting $j_2 \leq j_1 \leq j$, in which case the number of distinct bispectrum components drops to $(J+1)(J+2)(J+\frac{3}{2})/3$, a three-fold reduction in the limit of large $J$.

## 2.2 SNAP Potential Energy Function

Given the bispectrum components as descriptors of the neighborhood of each atom, it remains to express the potential energy of a configuration of $N$ atoms in terms of these descriptors. We write the energy of the system containing $N$ atoms with positions $\mathbf{r}^N$ as the sum of a reference energy $E_{ref}$ and a local energy $E_{local}$

$$E(\mathbf{r}^N) = E_{ref}(\mathbf{r}^N) + E_{local}(\mathbf{r}^N). \tag{2.8}$$

The reference energy includes known physical phenomena, such as long-range electrostatic interactions, for which well-established energy models exist. $E_{local}$ must capture all the additional effects that are not accounted for by the reference energy. Following Bartok et al.[5, 4] we assume that the local energy can be decomposed into separate contributions for each atom,

$$E_{local}(\mathbf{r}^N) = \sum_{i=1}^{N} E_i(\mathbf{q}_i) \tag{2.9}$$

where $E_i$ is the local energy of atom $i$, which depends on the set of descriptors $\mathbf{q}_i$, in our case the set of $K$ bispectrum components $\mathbf{B}^i = \{B^i_1, \ldots, B^i_K\}$. The original GAP formulation of Bartok et al.[5] expressed the local energy in terms of a Gaussian process kernel. For the materials that we have examined so far, we have found that energies and forces obtained from quantum electronic structure calculations can be accurately reproduced by linear contributions from the lowest-order bispectrum components, with linear coefficients that depend only on the chemical identity of the central atom:

$$E^i_{SNAP}(\mathbf{B}^i) = \beta_0^{\alpha_i} + \sum_{k=1}^{K} \beta_k^{\alpha_i} B^i_k = \beta_0^{\alpha_i} + \boldsymbol{\beta}^{\alpha_i} \cdot \mathbf{B}^i \tag{2.10}$$

where $\alpha_i$ is the chemical identity of atom $i$ and $\beta_k^{\alpha}$ are the linear coefficients for atoms of type $\alpha$. Hence the problem of generating the interatomic potential has been reduced to that of choosing the best values for the linear SNAP coefficients. Since our goal is to reproduce the accuracy of quantum electronic structure calculations for materials under a range of conditions, it makes sense to select SNAP coefficients that accurately reproduce quantum calculations for small configurations of atoms representative of these conditions.

In quantum methods such as density functional theory[21] the most readily computed properties are total energy, atom forces, and stress tensor. The linear form of the SNAP energy allows us to write all of these quantities explicitly as linear functions of the SNAP coefficients. We restrict ourselves here to the case of atoms of a single type, but the results are easily extended to the general case of multiple atom types. In the case of total energy, the SNAP contribution can be written in terms of the bispectrum components of the atoms

$$E_{SNAP}(\mathbf{r}^N) = N\beta_0 + \boldsymbol{\beta} \cdot \sum_{i=1}^{N} \mathbf{B}^i \tag{2.11}$$

where $\boldsymbol{\beta}$ is the $K$-vector of SNAP coefficients and $\beta_0$ is the constant energy contribution for each atom. $\mathbf{B}^i$ is the $K$-vector of bispectrum components for atom $i$. The contribution of the SNAP energy to the force on atom $j$ can be written in terms of the derivatives of the bispectrum components w.r.t. $\mathbf{r}_j$, the position of atom $j$

$$\mathbf{F}^j_{SNAP} = -\nabla_j E_{SNAP} = -\boldsymbol{\beta} \cdot \sum_{i=1}^{N} \frac{\partial \mathbf{B}^i}{\partial \mathbf{r}_j}, \tag{2.12}$$

where $\mathbf{F}^j_{SNAP}$ is the force on atom $j$ due to the SNAP energy. Finally, we can write the contribution of the SNAP energy to the stress tensor

$$\mathbf{W}_{SNAP} = -\sum_{j=1}^{N} \mathbf{r}_j \otimes \nabla_j E_{SNAP} = -\boldsymbol{\beta} \cdot \sum_{j=1}^{N} \mathbf{r}_j \otimes \sum_{i=1}^{N} \frac{\partial \mathbf{B}^i}{\partial \mathbf{r}_j} \tag{2.13}$$

where $\mathbf{W}_{SNAP}$ is the virial tensor due to the SNAP energy and $\otimes$ is the Cartesian outer product operator.

All three of these expressions consist of the vector $\boldsymbol{\beta}$ of SNAP coefficients multiplying a vector of quantities that are calculated from the bispectrum components of atoms in a configuration. This linear structure greatly simplifies the task of finding the best choice for $\boldsymbol{\beta}$. We can define a system of linear equations whose solution corresponds to an optimal choice for $\boldsymbol{\beta}$, in that it minimizes the sum of square differences between the above expressions and the corresponding quantum results defined for a large number of different atomic configurations. This is described in more detail in the following section.

## 2.3 Formulation of the Linear Least Squares Problem

The previous section outlined the SNAP formulation. In practice, one needs to determine the values of the SNAP coefficients, $\boldsymbol{\beta}$. This section presents how we solve for the $K$-vector $\boldsymbol{\beta}$ of SNAP coefficients using a least-squares formulation. The fitting problem is overdetermined, in the sense that the number of data points that we are fitting to far exceeds the number of SNAP coefficients. The cost of evaluating the bispectrum components $B_{j_1,j_2,j}$ increases strongly with the order of the indices $j$, $j_1$, and $j_2$. For this reason, $K$ is limited to the range $10-100$. In contrast, with the availability of high-performance computers and highly optimized electronic structure codes, it is not difficult to generate data for hundreds or thousands of configurations of atoms. Note that we refer to a *configuration* as a set of atoms located at particular positions in a quantum mechanical calculation. In most cases, the atoms define an infinite repeating structure with specified periodic lattice vectors. For a particular configuration $s$, containing $N_s$ atoms, the electronic structure calculation yields $3N_s+7$ data values: the total energy, the $3N_s$ force components, and the 6 independent components of the stress tensor. The same quantities are calculated for the reference potential. In addition, the bispectrum components and derivatives for each atom in the configuration are calculated. We can use all of this data to construct the following set of linear equations.

$$\begin{bmatrix} \vdots & \vdots \\ N_s & \sum_{i=1}^{N_s} \mathbf{B}^i \\ \vdots & \vdots \\ 0 & -\sum_{i=1}^{N_s} \frac{\partial \mathbf{B}^i}{\partial r_j^\alpha} \\ \vdots & \vdots \\ 0 & -\sum_{j=1}^{N_s} r_j^\alpha \sum_{i=1}^{N_s} \frac{\partial \mathbf{B}^i}{\partial r_j^\beta} \\ \vdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} \beta_0 \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \vdots \\ E_s^{qm} - E_s^{ref} \\ \vdots \\ F_{j,\alpha}^{qm} - F_{j,\alpha}^{ref} \\ \vdots \\ W_{\alpha\beta,s}^{qm} - W_{\alpha\beta,s}^{ref} \\ \vdots \end{bmatrix} \tag{2.14}$$

This matrix formulation is of the type $\mathbf{A}\cdot\boldsymbol{\beta} = \mathbf{y}$ which can be solved for the coefficients $\boldsymbol{\beta}$. The

optimal solution $\hat{\boldsymbol{\beta}}$ for this set of equations is [33]:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|(\mathbf{A} \cdot \boldsymbol{\beta} - \mathbf{y})\|^2 = \mathbf{A}^{-1} \cdot \mathbf{y} \tag{2.15}$$

In practice, we do not explicitly take the inverse of the $\mathbf{A}$ matrix, but instead use a QR factorization to solve for $\boldsymbol{\beta}$. We have found the linear solve to obtain the optimal SNAP coefficients to be very fast and not poorly conditioned.

We have also added the capability to perform weighted least squares to weight certain rows more than others. That is, we add a vector of weights $\mathbf{w}$ to the formulation of the minimization formulation:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{w} \circ (\mathbf{A} \cdot \boldsymbol{\beta} - \mathbf{y})\|^2 \tag{2.16}$$

where $\circ$ is used to denote element by element multiplication by the weight vector. Thus, each row in the $\mathbf{A}$ matrix and the $\mathbf{y}$ vector are multiplied by a weight specified for that row. In this way, we are able to specify weights per configuration type (e.g. BCC crystals, liquids, etc.) and per quantity of interest (e.g. energy, force, virial). We have found that the ability to weight different rows in the $\mathbf{A}$ matrix is critical to ensure the regression works well. One reason is that the total energy, forces, and stress components can vary considerably in relative magnitude, depending on what units they are expressed in. However, the more important reason is that it is desirable to control the relative influence of different configurations, depending on the material properties that are of greatest importance. For example, if we want the SNAP potential to more accurately reproduce BCC elastic constants, we can increase the weight on the stress components of strained BCC configurations. We also found it helpful to convert all extensive quantities to intensive quantities, in order to counteract overweighting of configurations with large $N_s$. Total energy rows were scaled by the number of atoms and virial components rows were scaled by the cell volume.

# Chapter 3

# SNAP Fitting Software in LAMMPS

We describe the framework to generate the SNAP fit within the LAMMPS software tool. We start with quantum mechanical (QM) training data, generated from ab initio calculations. To obtain the training data, we perform density functional theory (DFT) simulations for relevant configurations. The training data is populated with information such as atom coordinates, atom types, the cell matrix, energies, forces, and virials. The training data must be converted to JSON format. Note that the performance of the SNAP potential will depend on the comprehensiveness of the configurations in the training set. In the example of tantalum below, we are interested in material plasticity and stability as well as elastic constants and the lattice parameter. The QM training data included configurations with various crystal structures, energy-volume, generalized stacking fault, free surface, and liquid structures.

We have developed a robust, Python-based tool to generate SNAP potentials from training set data based on the weighted linear regression shown in Equation 2.16. Named fitsnap.py and described in the User's Manual included as Appendix A, it has a number of features to simplify and reduce the time needed to generate a SNAP potential. Some of these include a flexible input syntax, use of the efficient and widely available linear algebra package NumPy, parallel operation, and caching of intermediate results for later reuse.

When run, fitsnap.py first converts JSON-format training set data into a configuration format understood by LAMMPS. It then invokes LAMMPS, which generates the bispectrum components for the training data configurations. LAMMPS also generates a ZBL repulsive core potential which serves as the reference potential. This potential is added to the SNAP potential for the total potential, as explained in Section 2.2 above.

After LAMMPS generates the bispectrum components, fitsnap.py parses its output to collect, aggregate, and sum the appropriate quantities to create the $A$ matrix defined in Equation 2.14. The output quantities describe the $y$ vector in $A\beta = y$, where $y$ includes total and reference energies, forces, and stress tensors. The fitsnap.py script sends the $A$ matrix, the $y$ vector, and the associated weights to a least-squares solver which generates the optimal regression coefficients, $\hat{\beta}$. The resulting coefficients along with several hyperparameters make up the SNAP potential, which is written out for later use in LAMMPS.

The steps involved in the generation of a SNAP potential are shown in the red box (generation of training data), orange box (Python script fitsnap.py), and blue box (LAMMPS) on the right side of Figure 3.1.

## 3.1 Optimization of hyperparameters governing the SNAP potential

The steps involved in generating a SNAP potential involve the user specification of several "hyperparameters" such as the weights for each training configuration. In addition, the generation of the bispectrum components is governed by several parameters that the user can specify, including the highest order terms to include in the bispectrum component calculation and the cutoff distance defining the neighborhood of an atom in the SNAP bispectrum calculation. It is not obvious how to choose values of these hyperparameters which will lead to an "optimal" SNAP potential in the sense of minimizing the SNAP prediction error with respect to energy errors, force errors, or other quantities.

To determine the optimal hyperparameters governing the SNAP potential, we have used an optimization framework and placed it around the SNAP calculation to generate many instances of SNAP potentials. The optimization framework we use is the DAKOTA software [2], which is a toolkit of optimization and uncertainty quantification methods designed to interface to scientific computing codes. The process of generating a SNAP potential within an optimization loop to optimize the governing parameters is shown in Figure 3.1. DAKOTA varies input parameters such as the weights per configuration group and the cutoff distance. These values are then specified in the SNAP generation and the SNAP potential is calculated. Once the SNAP coefficients are generated, they are used to predict the energy and forces of the QM training data. The errors in the SNAP prediction (defined as the difference between SNAP configuration energies vs. QM configuration energies, SNAP forces vs. QM forces, etc.) are then aggregated into an objective function which is returned to DAKOTA and used as the quantity which DAKOTA tries to optimize.

Using this optimization framework, we can identify the SNAP potential that has the "best" result, according to minimizing an objective function. We have examined various objective functions. Currently the objective function we use involves a weighted sum of the errors with respect to energies, forces, and virials, as well as errors with respect to the elastic constants. The objective function is not trivial to define: the resulting SNAP potential can be very sensitive to which error measures are weighted more in the objective function.

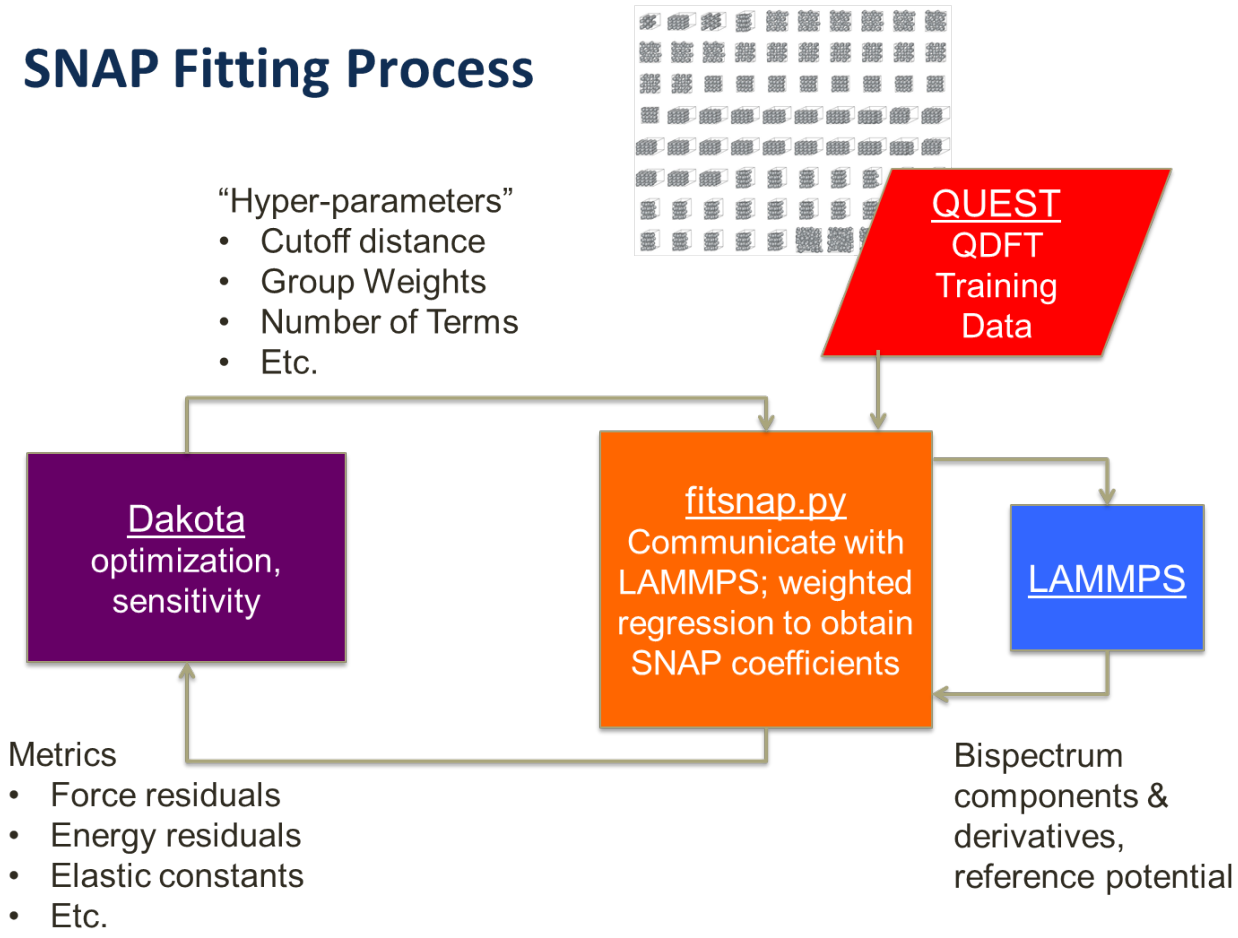


**Figure 3.1.** Flowchart of the SNAP Fitting process: the optimization loop coupled with the generation of the SNAP potential in LAMMPS

# Chapter 4

# SNAP Potential for Tantalum

## 4.1 Training Data

The training set data as well as the validation data for these potentials were computed using density functional theory (DFT) electronic structure calculations as implemented in the Vienna Ab initio Simulation Package (VASP)[16]. A variety of different types of atomic configurations were used to construct the full set of training data, and these are summarized in Table 4.1. Configurations of different types were chosen to adequately sample the important regions of the potential energy surface. Configurations of type "Displaced" were constructed by randomly displacing atoms from their equilibrium lattice sites in supercells of the A15, BCC, and FCC crystal structures. Configurations of type"Elastic" were constructed by applying random strains to primitive cells of the BCC crystal. The configurations of type "GSF" consist of both relaxed and unrelaxed generalized stacking faults along the [110] and [112] crystallographic directions. The configurations of type "Liquid" were taken from a high-temperature quantum molecular dynamics simulations of molten tantalum. The configurations of type "Surface" consisted of relaxed and unrelaxed [100], [110], [111], and [112] BCC surfaces. For each type of configuration we specified a weight for the energy, force, and stress. We set the force and energy weights of the"Elastic" configurations to zero and we set the stress weights of all other configurations to zero.

In addition to the training data and way in which different quantities were weighted, the quality of the SNAP potential also was somewhat dependent on the choices made for the reference potential and the SNAP hyperparameters. Because the training data did not sample highly compressed configurations, it was important that the reference potential provide a good physical description of Pauli repulsion that dominates the interaction at close separation. We chose the Ziegler-Biersack-Littmark (ZBL) empirical potential that has been found to correctly correlate the high-energy scattering of ions with their nuclear charge $Z_{zbl}$[41]. Because the ZBL potential decays rapidly with radial separation, we used a switching function to make the energy and force go smoothly to zero at a distance $R_{zbl,o}$, while leaving the potential unchanged for distances less than $R_{zbl,i}$. The values for these three parameters are given in Table 4.2. For the SNAP basis functions, we used the same sinusoidal switching function as Bartok et al. [5].

$$f_c(r) = \frac{1}{2}(\cos(\pi r/R_{cut})+1), r \leq R_{cut} \quad (4.1)$$

$$= 0, r > R_{cut} \quad (4.2)$$

| Type | $N_{conf}$ | $N_{atoms}$ | Energy | Force | Stress |
|---|---|---|---|---|---|
| Displaced A15 | 9 | 64 | 100 | 1 | - |
| Displaced BCC | 9 | 54 | 100 | 1 | - |
| Displaced FCC | 9 | 48 | 100 | 1 | - |
| Elastic BCC | 100 | 2 | - | - | 0.0001 |
| GSF 110 | 22 | 24 | 100 | 1 | - |
| GSF 112 | 22 | 30 | 100 | 1 | - |
| Liquid | 3 | 100 | 100 | 1 | - |
| Surface | 7 | 30 | 100 | 1 | - |

**Table 4.1.** DFT data used to fit the SNAP potential for Tantalum

| | |
|---|---|
| $J$ | 3 |
| $R_{cut}$ | 4.67637 Å |
| $\theta_0^{max}$ | $0.99363\pi$ |
| $R_{zbl,i}$ | 4.0 Å |
| $R_{zbl,o}$ | 4.8 Å |
| $Z_{zbl}$ | 73.0 |

**Table 4.2.** Definition of SNAP ZBL potential.

The DAKOTA package was used to optimize the value of $R_{cut}$ so as to minimize the error in the energies, forces, and elastic constants relative to the training data. The resultant value of $R_{cut} = 4.67637$ is physically reasonable, as it includes the 14 nearest neighbors in the BCC crystal, and the first coordination shell in the melt. The effect of using fewer or more bispectrum components was examined experimentally by varying $J$. We found that the fitting errors decreased monotonically with increasing $J$, but the marginal improvement also decreased. We found that truncating at $J = 3$ provided a good tradeoff between accuracy and computational efficiency. The full set of ZBL and SNAP parameters values are given in Table 4.2, while the values of the SNAP linear coefficients corresponding to each bispectrum component are listed in Table 4.3

## 4.2 Validation Results

One of the crucial features of an interatomic potential model is that it predicts the correct minimum energy crystal structure and that the energetics of competing crystal structures are qualitatively correct. Figure 4.1 plots the energy per atom computed with the SNAP potential as a function of volume for the BCC, FCC, A15, and HCP phases. The energy of diamond structure Ta was also

| $k$ | $2j_1$ | $2j_2$ | $2j$ | $\beta_k$ |
|---|---|---|---|---|
| 0 | | | | -2.92477 |
| 1 | 0 | 0 | 0 | -0.01137 |
| 2 | 1 | 0 | 1 | -0.00775 |
| 3 | 1 | 1 | 2 | -0.04907 |
| 4 | 2 | 0 | 2 | -0.15047 |
| 5 | 2 | 1 | 3 | 0.09157 |
| 6 | 2 | 2 | 2 | 0.05590 |
| 7 | 2 | 2 | 4 | 0.05785 |
| 8 | 3 | 0 | 3 | -0.11615 |
| 9 | 3 | 1 | 4 | -0.17122 |
| 10 | 3 | 2 | 3 | -0.10583 |
| 11 | 3 | 2 | 5 | 0.03941 |
| 12 | 3 | 3 | 4 | -0.11284 |
| 13 | 3 | 3 | 6 | 0.03939 |
| 14 | 4 | 0 | 4 | -0.07331 |
| 15 | 4 | 1 | 5 | -0.06582 |
| 16 | 4 | 2 | 4 | -0.09341 |
| 17 | 4 | 2 | 6 | -0.10587 |
| 18 | 4 | 3 | 5 | -0.15497 |
| 19 | 4 | 4 | 4 | 0.04820 |
| 20 | 4 | 4 | 6 | 0.00205 |
| 21 | 5 | 0 | 5 | 0.00060 |
| 22 | 5 | 1 | 6 | -0.04898 |
| 23 | 5 | 2 | 5 | -0.05084 |
| 24 | 5 | 3 | 6 | -0.03371 |
| 25 | 5 | 4 | 5 | -0.01441 |
| 26 | 5 | 5 | 6 | -0.01501 |
| 27 | 6 | 0 | 6 | -0.00599 |
| 28 | 6 | 2 | 6 | -0.06373 |
| 29 | 6 | 4 | 6 | 0.03965 |
| 30 | 6 | 6 | 6 | 0.01072 |

**Table 4.3.** SNAP linear coefficients for tantalum.

computed. As expected, it was found to be substantially ($\sim$2.8 eV/atom) higher than the BCC phase and is not included on the plot. In addition, energies computed from density functional theory are included as crosses. It is seen that the relative energy of these phases is correctly predicted. The BCC phase is the most stable throughout the volume range considered, with the A15 phase somewhat higher. The FCC phase is next with a minimum energy about 0.2 eV/atom above that of BCC. Note that these energy differences are very consistent with the DFT calculations. The SNAP predicted HCP energy is also shown. Note that the SNAP potential is able to differentiate

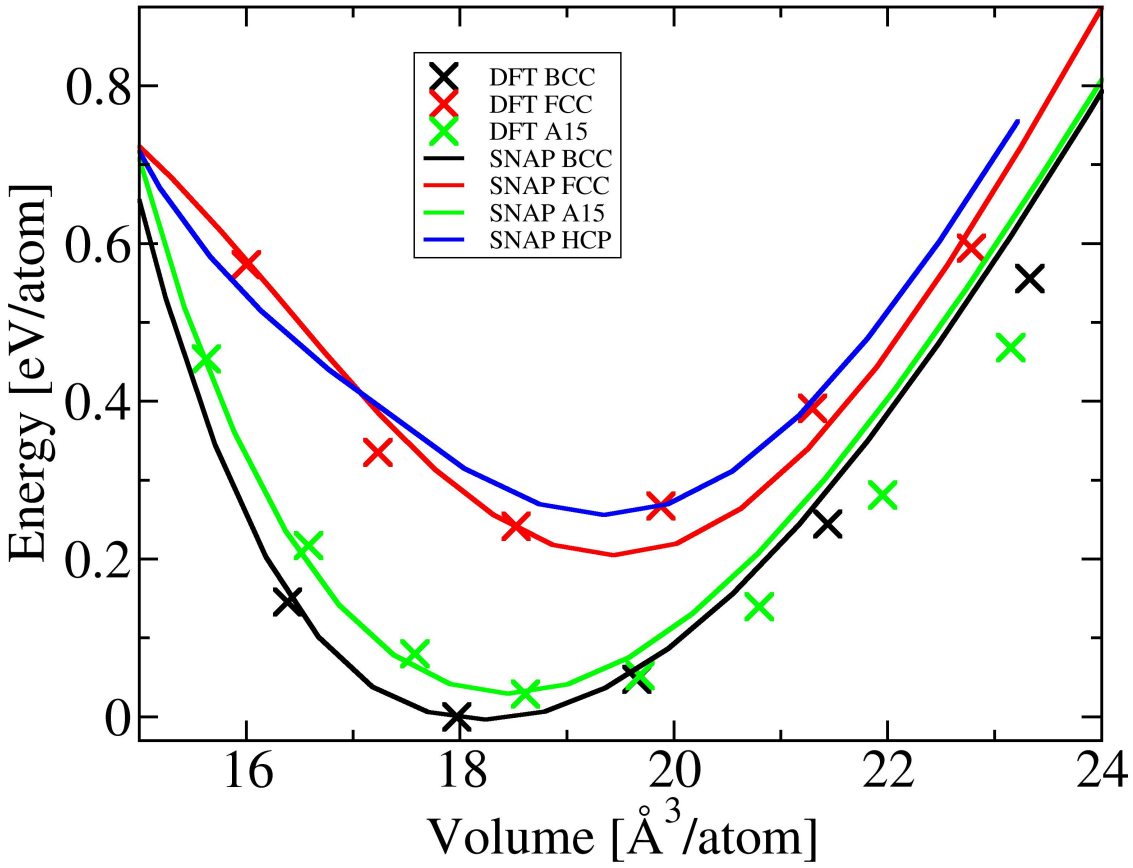


**Figure 4.1.** Energy versus volume for various crystal structures as indicated in the legend, as computed by SNAP (solid curves) and from DFT (x).

HCP and FCC crystal structures which are structurally very similar. The SNAP potential predicts that the HCP structure is higher in energy than the FCC structure. This is a prediction in that no HCP data was used in the potential construction. The relative energies of HCP and FCC are in agreement with our DFT calculations, which show that lowest energy HCP structure (not shown) lies 0.04 eV/atom above the minimum energy FCC structure. Further, the SNAP potential predicts the HCP $c/a$ ratio to be 1.72, which is considerably greater than ideal value $c/a \approx 1.63$. The DFT calculations for HCP (not shown) predict an even larger value of $c/a = 1.77$.

## 4.3 Melting point and Liquid structure

The melting point predicted by the SNAP potential has been determined. An atomistic slab was created and brought to temperature above the melting point using a Langevin thermostat until the

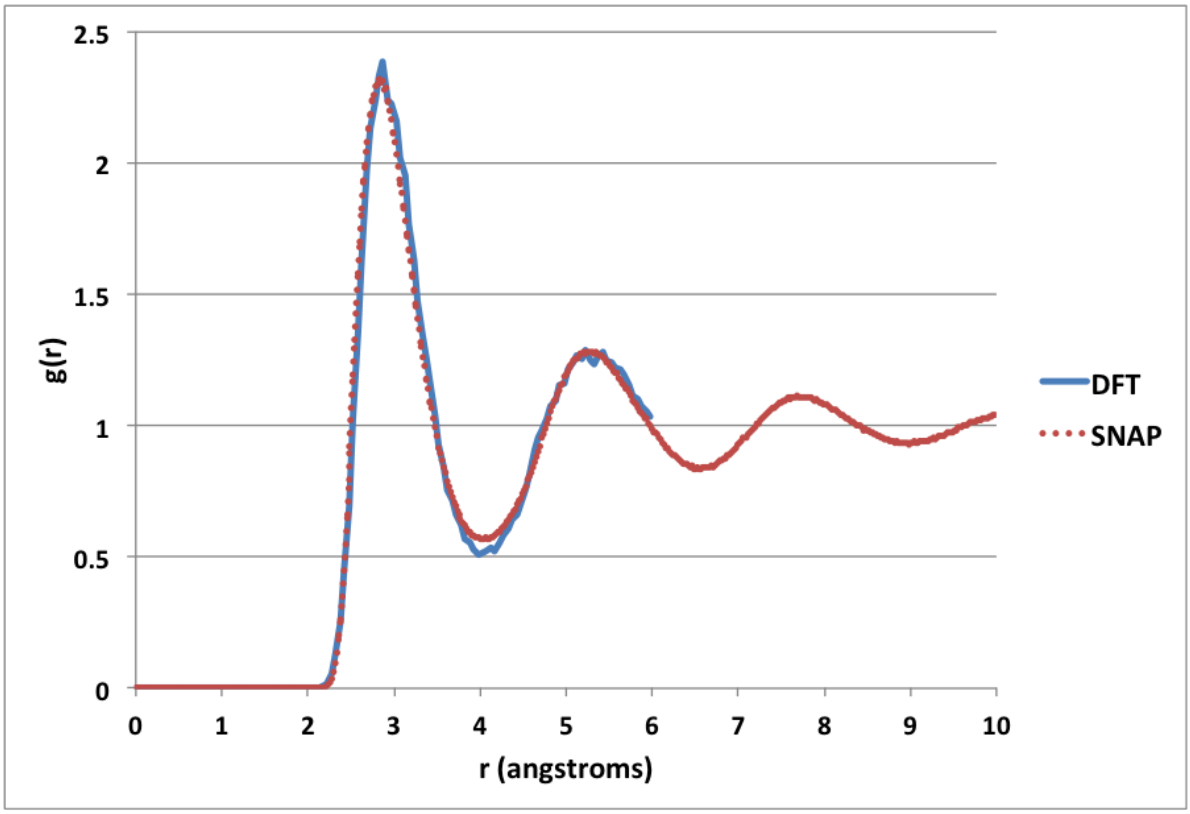


**Figure 4.2.** Comparison of pair correlation for molten tantalum calculated using DFT and SNAP

surface of the slab was melted. The molecular dynamics simulation was then continued in an NVE ensemble. This resulted in a system containing two solid-liquid interfaces. The temperature of the MD system now fluctuated around the equilibrium melting temperature. It is important that the solid phase be at the correct melting point density. This was ensured through a simple iterative procedure. An estimate of the melting point was obtained for an assumed lattice constant, the lattice constant of the solid at that temperature was determined from an NPT simulation of the solid, and the melting point was determined with the interfacial area determined by this lattice constant. This process was iterated until the assumed and predicted melting points agreed.

This procedure predicted a melting point of 2790 K. This value is in reasonable agreement with the experimental melting point of 3293 K. It should be noted that the melting point is typically a difficult quantity for interatomic potentials to predict accurately. Further, the comparison with experiment is not a direct test of the agreement of the SNAP potential with DFT calculations. The experimental value reflects contributions of the free energy of electronic excitations. Further, the DFT prediction for the melting point is not known.

While configurations that correspond to molten Ta were used in the training set, it is important to determine if the potentials actually reproduce the correct distribution of spatial density corre-

lations in the liquid state. The liquid is an important test of potential models since it samples configurations that are far from those of the equilibrium solid crystals. In particular, the liquid structure depends strongly on the repulsive interactions that occur when two atoms approach each other. We calculated the pair correlation function of the liquid, $g(r)$, both using DFT and from the SNAP potential. These simulations were performed for the same temperature, 3250 K, and atomic density of 49.02 atom/nm$^3$. The DFT simulation treated 100 atoms for a period of 2 ps while the SNAP simulations considered a cell containing 1024 atoms and averaged over 500 ps. Figure 4.2 compares $g(r)$ obtained in the two simulations. The agreement is excellent except perhaps in the region of the first minimum. Note that there is substantially more statistical uncertainty in the DFT result due to the short simulation time and the DFT structure cannot be determined beyond about 0.6 nm, due to the smaller simulation cell. These results indicate that the SNAP potential provides a good representation of the molten structure.

energy associated with surfaces, unstable stacking faults, vacancies and self-interstitial atoms. The results obtained from the SNAP potential are compared to our DFT calculations and also against two other interatomic potential models, the embedded atom method (EAM) model developed by Zhou et al.[40] and the angular dependent potential (ADP) due to Mishin and Lozovoi[24].

## 4.4 Dislocations

Figure 4.3 shows the screw dislocation migration energy barrier calculated using DFT and the SNAP, EAM, and ADP potentials. One of the dominant deformation mechanisms for metallic materials is the motion of dislocations. For the case of body-centered cubic materials such as tantalum, the screw dislocations are known to play a crucial role. The structure and motion of screw dislocations in BCC metals has been examined for many years and is discussed in detail by Gröger et al.[11] and by references therein. A crucial feature of screw dislocations in BCC metals is the Peierls barrier which is the energy barrier to move the dislocation to its next stable configuration. Unlike face-centered-cubic metals where the Peierls barrier is generally negligible, the barrier in the case of BCC metals is substantial and plays a significant role in mechanical deformation. As has been shown recently by Weinberger et al.[39], many empirical potentials for BCC metals predict qualitatively incorrect Peierls barriers. DFT calculations show that the Peierls is has a simple shape with a single hump while many empirical potentials predict a transition path with two maxima and a metastable intermediate state. The Peierls barrier computed via the SNAP potential is shown in Figure 4.3 along with the DFT barrier and the barriers predicted by the Zhou and ADP potentials. In all cases the barriers were computed based on a dislocation dipole configuration as described by Weinberger et al.[39]. While the ADP and Zhou potentials both predict incorrect barriers with an intermediate metastable state, the SNAP potentials predicts a barrier with a single maximum in agreement with the DFT results. Further, the magnitude of the barrier is in excellent agreement with the DFT prediction.

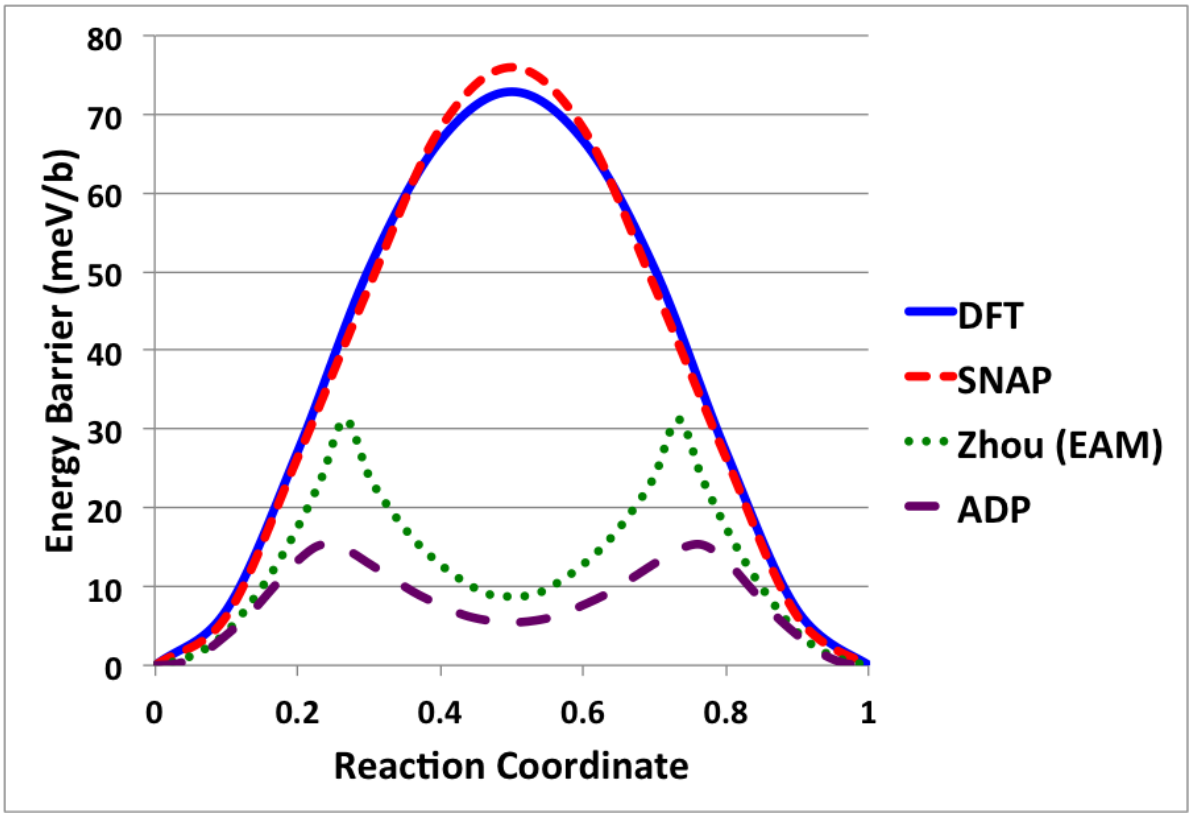


**Figure 4.3.** Comparison of screw dislocation migration energy barrier calculated using DFT and the SNAP, EAM, and ADP potentials.

# Chapter 5

# SNAP Potential for Indium Phosphide

In our original proposal for this project, we identified indium phosphide (InP) as the second material, after tantalum, for which we would develop quantum-accurate interatomic potentials. InP was chosen for several reasons. Firstly, it provides a good test for the ability of SNAP to distinguish chemical elements. Secondly, understanding the atomic-scale behavior of defects in crystalline InP is important in a variety of applications, as described below. Thirdly, InP is representative of a much larger class of III-V semiconductor compounds and alloys of compounds that are technologically important.

For InP, the goal is to develop a potential that can meaningfully represent the particle dynamics within a high-energy collision cascade. The target application requires the ability to characterize the nature and distribution of defects after a high-energy collision cascade. The demands upon an interatomic potential in such an application are severe, requiring the ability to faithfully reproduce dynamics at high interaction energies and forces, and also requiring good accuracy in computing the formation energies of the ultimate displacement damage defects that appear in a cascade. For GaAs, a Pettifor bond-potential [27] proved adequate for cascade simulations in GaAs, but trial interatomic potentials have, thus far, been unable to adequately reproduce the chemistry of InP and other III-V alloys. The intent here is to establish a protocol to create effective SNAP potentials when needed, enabling dynamical simulations of cascades in arbitrary III-V materials. The first step is to treat a pure binary compound composed of just two elements. Ultimately it is necessary to treat the ternary and quaternary compounds of technological interest.

To develop a SNAP potential for a binary compound such as InP requires generalizing the SNAP potential to capture additional complexity that does not arise in the case of a pure element, where all atoms have the same chemical identity. In elemental tantalum, the SNAP potential need only keep track of the distribution of particles in the local environment of each atom. In InP, a successful potential must be able to distinguish between the different elemental components; the chemistry around each atom is not just dictated by the positions of nearby particles, but also the elemental identity of those particles. Within the simple form of SNAP, the means to do this is not uniquely defined. As described later, we designed and implemented a series of discriminating aspects into SNAP that do distinguish between different elements. The initial step in the process is to develop a training set of data that adequately spans the multicomponent chemistry that is desired.

## 5.1 InP training set development

The training set data for InP for the SNAP potentials was composed of small structures, for which energy, forces, and stress tensors were computed using density functional theory (DFT) electronic structure calculations. For InP, the DFT calculations were performed with Sandia's SeqQuest DFT code (http://dft.sandia.gov/Quest/), using atomic pseudopotentials that placed the indium 4*d* electrons into the core (treating In as a trivalent atom, within the local density approximation [29]. Additional computational details can be found in a SAND Report that comprehensively described all simple intrinsic defects using DFT [31]. The specific atomic pseudopotential and DFT functional are not particularly crucial, and other choices might arguably have greater accuracy. For the development of the training set, the important aspect is to have a consistent computational context throughout, and to have a training set that reasonably accurately spans the chemical environments that are potentially of interest in simulations.

The first stage of SNAP development for InP began with a training set composed of a collection of canonical structures and some extensions intended to yield a comprehensive set of chemical environments in a process that could potentially be automated. The canonical structures consisted of a wide sampling of binary stoichiometric crystal structures: the ground state zincblende (B3), the hexagonal wurzite (B4), ionic B1 and B2, a binary form of the B8 structure (two types), multiple forms of the cinnabar (B9) structure, a binary forms of the bct-$C4$ , bct5, $\beta$-tin, hcp (two forms) structures, and $SC16$ structures. This explored a variety of coordination numbers, bond angles, and strain. The pure-phosphorus bulk crystal structures were omitted, because conventional DFT does poorly for discriminating the ground state P crystalline structures. For balance, the pure-In bulk crystal structures were omitted as well. Each of these crystals was generated at ambient pressure, and then both isotropically compressed in 4 GPa increments to 40 GPa, and expanded in 2 GPa increments to -10 GPa, the stability limit under tension. The rationale was both to vary the local coordination of both elements and also to probe variations of bond lengths and particle densities. To this was added a series of 8-atom cluster calculations, with pure $In_8$ and $P_8$ cubes, and a series of $In_4P_4$ clusters with the In and P atoms decorating the corners of cube,

The extensions involved a series of bulk samples, designed to encompass the space of somewhat randomized structures that would appear in the damaged regions of a collision cascade in bulk InP. These bulk samples were represented by 216 atom cubic supercells with periodic boundary conditions in all three dimensions. A series of molten InP structures were generated by MD simulation using a crude analytic In-P interatomic potential. The atoms in the liquid samples ideally explore a variety of local chemical environments, with different bonds, coordinations, distances, and most importantly, defects. The quenched amorphous samples explore near-equilibrium structures with strains. These molten structures were then computed again with DFT, and the DFT energies, forces, and stress tensors were put into the training set. In addition, each of these structures was then relaxed ("quenched") to a local minimum energy structure using DFT. The DFT results for the relaxed structures, and a sampling of intermediate configurations were also placed in the training set. This completed the first stage. Zincblende crystal defect structures were intentionally withheld from this training set, in order to test the efficacy of the first stage protocol.

This first stage training set proved inadequate. The SNAP fit to this data was tested in a bulk

melt-and-quench MD simulation using a barostat, and produced an amorphous In-P structure much denser than nature, and lower in energy than the perfect crystal. In the second stage, the training set was augmented by DFT calculations of this dense MD structure and intermediate configurations from its relaxation. This was done first at the MD volume, then at the expected volume (linear dimensions scaled by 1.08) and at a larger volume (scaled by 1.16). Similarly, a molten InP structure from the first stage was both compressed and expanded about its equilibrium volume, to provide further training data bounding the expected volume.

An MD simulation with the resulting SNAP fit now produced the correct volume, but generated a new error, forming unphysical tight clusters of phosphorus atoms. Somehow the MD simulation found a path to allow phosphorus atoms to come to close approach and find an unphysical low-energy structure. To address this failure, the third stage of the training set added a series of "tight defect" structures to the training set. These contained a P antisite in the bulk InP zincblende crystal, i.e. a phosphorous atom replacing an indium atom in the lattice. To bias against tight clustering of P atoms, the sample first included the relaxed P antisite, and then a series of configurations where all four P neighbors moved inwards strongly or just one P-P bond was shortened, up to 1 eV/bond higher in energy. For balance, similar structures with In were added to the training set.

This third stage cured the tight clustering problem in the MD simulation. However, the calculated defect formation energies were inaccurate. In the fourth stage, the suite of 23 neutral defect structures, up to this point withheld from the process, was added to the training set. The resultant potential is referred to as Candidate 13 below. Again, the calculated defect formation energies were somewhat disappointing. The formation energies of the DFT structures were too low, and after relaxing these with SNAP, the defect formation energies dropped even more. However, subsequent candidate potentials which explored different variations of fitting, such as exploring different weighting for different aspects of the training set, significantly improved both the unrelaxed and relaxed defect formation energies. This is described in more detail below.

## 5.2 InP SNAP development

Indium phosphide required us to generalize the SNAP potential so that we are able to differentiate elements. Specifically, with InP, each atom can be either indium or phosphorous. These atoms are of different size and exhibit different chemistries and behaviors in any given local environment. The original SNAP potential was defined for a pure elemental material where no distinction between atom types was necessary.

We made distinctions between the In and P atoms by using different neighbor density weights, by defining element-specific linear coefficients and their bispectrum component contributions, and by distinct neighbor cutoffs for the different elements. Each of these proved necessary to achieve a reasonably accurate SNAP candidate for simulating InP. We have included these new features in the description of the SNAP potential in Section 2.1.

Ultimately, with the failure to produce a viable candidate without including the bulk defects

structures in the training set, accurately matching these structures became the emphasis of fitting the later SNAP candidates. We are interested in defect formation energies, but it is not sufficient to just match the DFT-computed defect formation energies of the training defects. The defect structures in the training set are fully relaxed according to DFT i.e. energy is a local minimum. But the structures are not relaxed according to the SNAP potential, unless the SNAP potential perfectly reproduced the DFT potential energy surface in the vicinity of the defect structures). When we relaxed the defect structure using the SNAP potential, we found that some SNAP candidates underwent significant further relaxation. The appropriate figure of merit for the intended application is how closely the relaxed defect energy with SNAP matches the relaxed defect energy in DFT. While the two relaxed structures might differ somewhat, the goal is to get relaxed energies that are the same. It should be noted that the symmetry of the bulk crystal guarantees that any reasonable potential will not allow an further relaxation of the bulk crystal. Hence, relaxation of the defect structures systematically lowers all the defect formation energies by some amount.

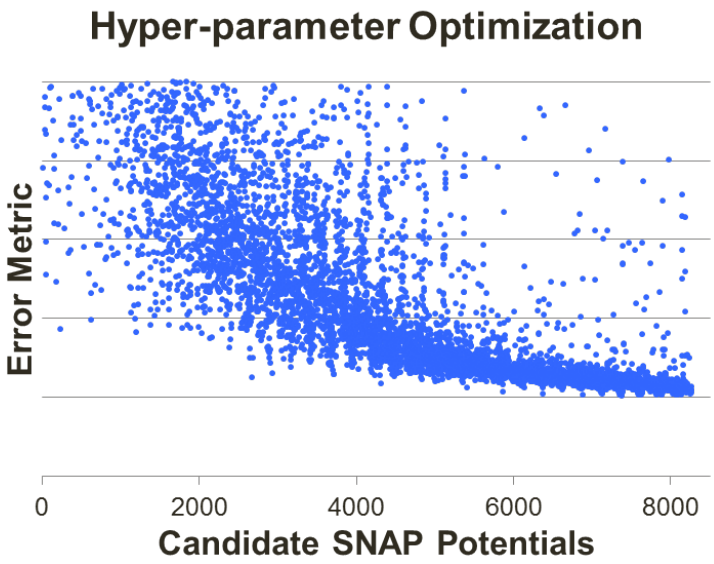


**Figure 5.1.** Example of a Hyperparameter optimization of the SNAP InP potential. 19 hyperparameters are being optimized, driving down the objective as the number of candidates examined increases.

We spent a significant amount of time optimizing the SNAP potential so that it would produce accurate relaxed defect energies. We used the genetic algorithm in the DAKOTA toolkit to perform the optimization. An example of this optimization is shown in Figure 5.1, where the optimization is performed over 19 hyperparameters governing the SNAP potential. These hyperparameters include the element radii, and weights for the energies and forces from various groups or configurations. The objective function being minimized in this figure combines the energy residuals (differences between SNAP and QM) for all configurations, force residuals for all configurations, and error in the defect formation energy, both for unrelaxed and relaxed structures. So for each "dot" in Figure 5.1, we run LAMMPS on approximately 350 structures, do the regression to obtain the

SNAP coefficients, then use the SNAP potential to relax the defect structures.

Figure 5.2 shows the representative results for a sensitivity analysis. We varied the linear regression weight given to the defect structure group while holding the weights of other groups constant at their nominal values. We plot the variation of four different quality metrics as the defect group weight is varied from very small to very large. "Energy Error" is the sum of the energy residuals for all configurations. "Force Error" is the sum of the force residuals for all configurations. "MSD" is the mean square displacement over all atoms during relaxation of structures in the defect group. "Relaxation" is the sum of the relaxation energies of the defect group structures. One can see from this figure that the force error remains relatively constant as the defect group weight is varied, but the overall energy error of all configurations increases while the error of the relaxation energies of the defect group decreases as the defect weight increases. This type of sensitivity analysis can show us what the trade-offs are and what values of the defect weights we might use (around $10^4$).

Figure 5.3 shows the defect formation energies calculated with two different SNAP potentials. The one in red, Candidate 13, has parameters that were hand-tuned, given our knowledge of the defect structures. The one in blue is the one produced by the optimization discussed in Figure 5.1. The comparison of these two potentials shows that using DAKOTA to optimize the hyperparameters results in a potential with significantly lower energy error. While the optimized SNAP potential provides a good overall fit to most of the groups, the defect energy error is still greater than 1 eV for most of the defect structures. For the intended application of predicting radiation effects this level of accuracy is marginal. The accuracy for defect formation energies is nonetheless superior to all existing published potentials, and for this reason, the new SNAP potential is undergoing further evaluation.

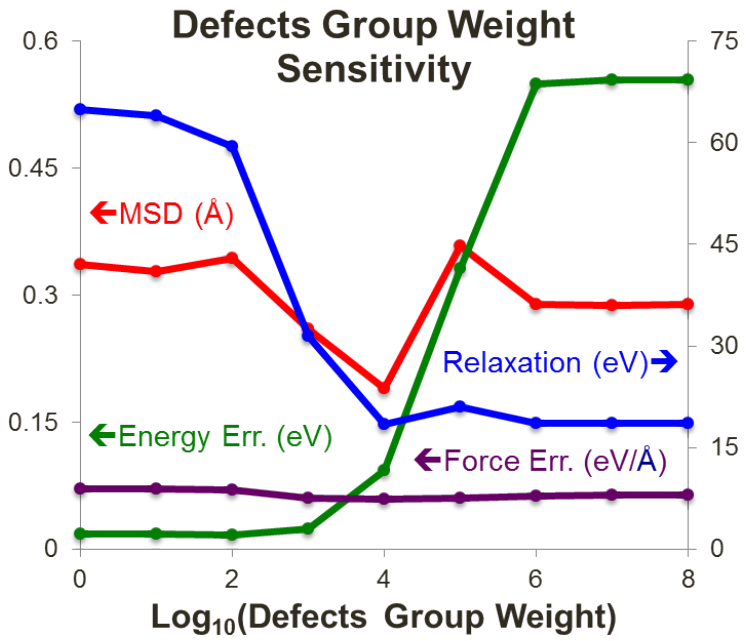


**Figure 5.2.** Sensitivity analysis of different metrics as the weight on the defect group increases in the SNAP InP potential.

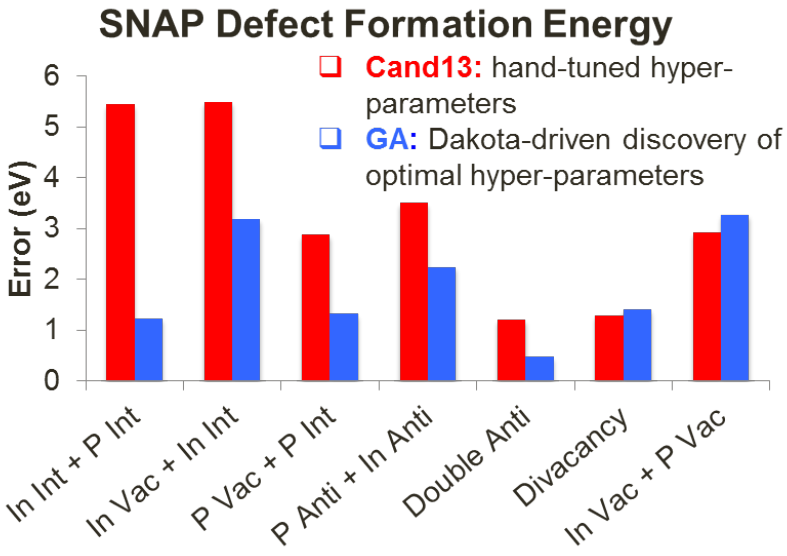


**Figure 5.3.** Errors in SNAP Defect Formation Energy for a variety of defect structures in Indium Phosphide

# Chapter 6

# SNAP Potential for Silica

As the reader is undoubtedly aware, SiO2 is a tremendously important material in many modern applications, including in the semiconductor industry. It would be of great value to have a quantum-accurate potential available that could model SiO2 in atomistically correct detail in all of its phases and polymorphs, as well as at the industrially-significant Si-SiO2 interface. While many SiO2 potentials exist, none are fully up to this challenge. We have attempted to develop a quantum accurate SNAP potential for SiO2 using the procedures outlined in this report. Results have been mixed, with some early successes noted, but with much more work still to be done. This chapter details our first attempts at developing a SNAP potential for SiO2, and presents some preliminary results for seven SiO2 crystal polymorphs and high-temperature liquid SiO2.

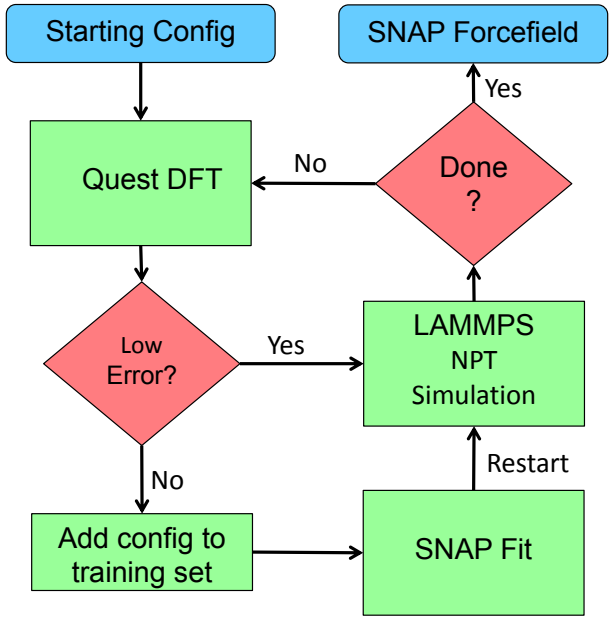


**Figure 6.1.** Conceptual diagram of the automated fitting procedure used for silica.

The training set for silica was generated on the fly by the following procedure. A LAMMPS simulation was run using SNAP, and forces were periodically checked by comparing to DFT. If the force error was low, then the number of timesteps before checking again was increased, and

the simulation was allowed to continue running. If the force error was high, then that particular configuration was added to the training set, and the simulation was restarted. A conceptual diagram of one possible way of implementing this procedure is shown in Figure 6.1. In this manner, the SNAP potential is generated on the fly (with less expert knowledge needed).

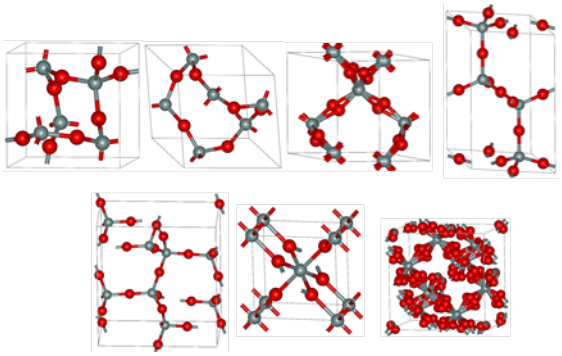

**Figure 6.2.** Crystal polymorph configurations used as part of the training data for the SNAP silica potential.

For seven SiO2 crystal polymorphs (see Figure 6.2), the SNAP potential shows very close agreement (less than 3% error) in the predicted lattice parameters as compared to the lattice constants generated by the SeqQuest QM data (see Figure 6.3). This indicates that the SNAP potential does a good job representing the equilibrium solid state for at least these SiO2 crystal polymorphs. We note that good agreement in this regard is relatively easy for a potential to achieve since all of the atoms are near potential energy minima and do not stray far from those potential wells. A much more difficult test is one that requires closer atomic approaches as would be seen in a high temperature liquid state.

We tested our new SNAP silica potential at two high temperature (3120 K and 3700 K) liquid states and compared the resulting radial distribution functions (RDFs, or g(r)) for O-Si, O-O, and Si-Si distances against the corresponding DFT-computed RDFs from the literature (see Ref. [15]) as shown in Figure 6.4. The agreement is remarkably good, indicating that our new SNAP SiO2 potential adequately captures the correct structure of liquid SiO2 at these temperatures. The slight differences between the 3120 K and the 3700 K RDF curves are nicely mimicked by the SNAP potential, indicating that the temperature dependency of the RDF is likewise being captured by the SNAP SiO2 potential.

While these preliminary results for bulk crystalline and molten states are promising, these do not guarantee that SNAP will perform well for more complicated structures. As we saw in the case of indium phosphide, matching DFT defect formation energies to within 1 eV is quite a stringent requirement. We have not yet tested SNAP's ability to predict defect formation energies.

Another very interesting analysis would be to test the ability of SNAP SiO2 to describe the structure of Si-SiO2 interfaces. The preliminary SNAP SiO2 potential described here did not include explicit point charge electrostatic interactions, so it would be difficult for it to capture

the known effect of charge redistribution across the Si-SiO2 interface. We could address this by combining SNAP with a variable charge reference potential. This approach is already used by several existing interatomic potentials for the Si-SiO2 interface, such as COMB and ReaxFF.

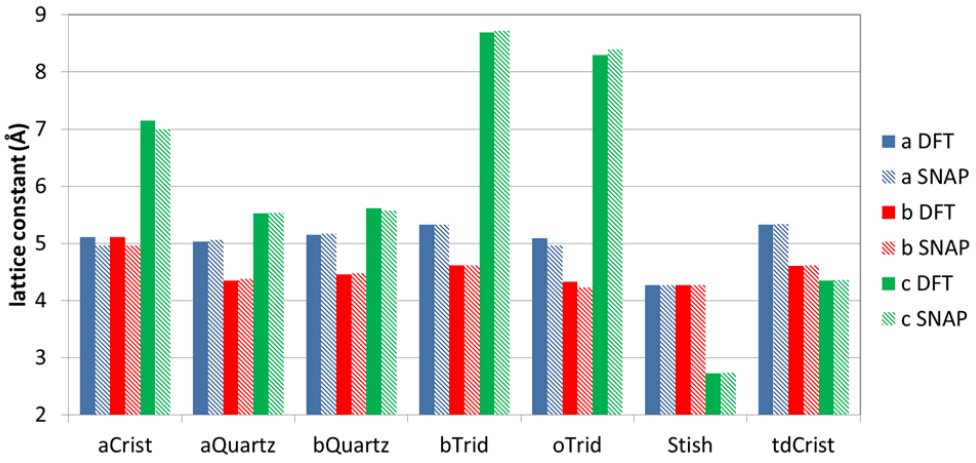


**Figure 6.3.** Agreement between QM results and SNAP estimates of lattice constant for various crystal polymorph configurations in silica.

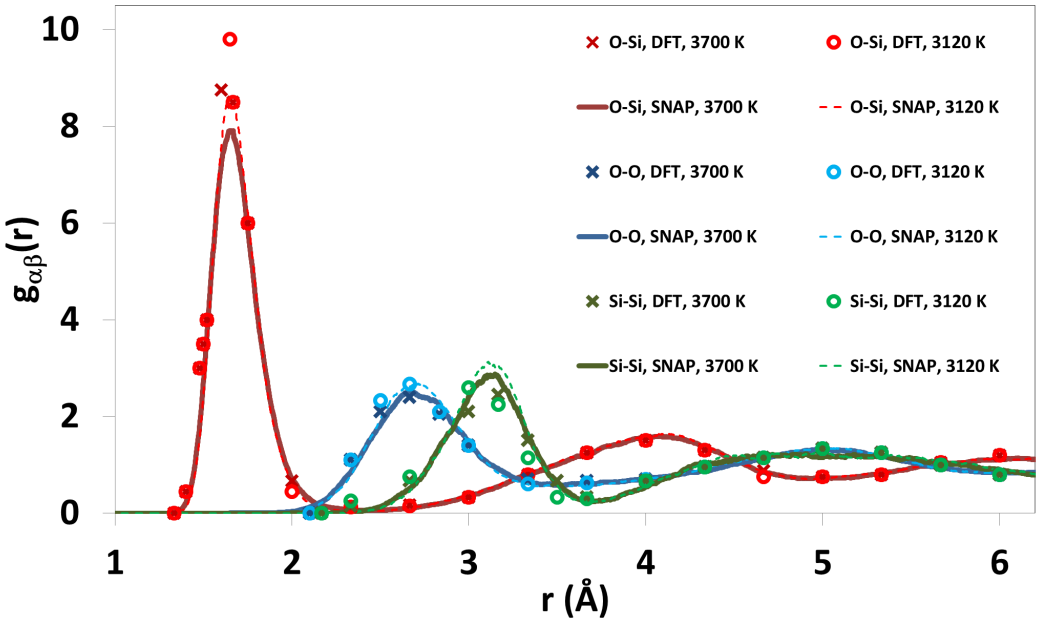


**Figure 6.4.** Agreement between QM results and SNAP estimates for liquid silica.

# Chapter 7

# Scaling Studies

One possible drawback of the SNAP methodology is computational cost when used in large-scale atomistic simulations. The number of floating point operations required by SNAP is one to two orders of magnitude greater than simpler potentials, such as EAM. At the same time, the communication requirements of SNAP remain similar to those of simpler potentials. These two characteristics make the SNAP potential suitable for running moderate sized scientific calculations on very large computers. Typical LAMMPS molecular dynamics (MD) simulations do not scale once the workload drops below several hundred atoms per processor, due to load imbalance and communication. With SNAP, we can achieve good scaling even when the workload drops below one atom per processor. To exploit this unprecedented strong scaling even further, we have developed specialized algorithms for load-balancing, thread parallelism, and GPU execution. At the same time, we have exploited a mathematical symmetry in the bispectrum components to reduce the computational cost of the force calculation. In this chapter, we summarize these algorithmic advances in scaling and single-node performance. The reader is referred to References [35] and [36] for more details.

The force on each atom due to the SNAP potential is formally given by Eq. 2.12. In order to perform this calculation efficiently, we use a neighbor list, as is standard practice in the LAMMPS code [19, 30]. This list identifies all the neighbors of a given atom $i$. In order to avoid negative and half-integer indices, we have switched notation from $u^{j}_{m,m'}$ to $u^{\eta}_{\mu,\mu'}$, where $\eta = 2j$, $\mu = m + j$, and $\mu' = m' + j$. Analogous transformations are used for $H^{jmm'}_{\substack{j_1 m_1 m'_1 \\ j_2 m_2 m'_2}}$ and $B_{j_1,j_2,j}$. This also allows us to reclaim the symbol $j$ for indexing the neighbor atoms of atom $i$. Finally, boldface symbols with omitted indices such as $\mathbf{u}_i$ are used to indicate a finite multidimensional arrays of the corresponding indexed variables.

Fig. 7.1 gives the resulting force computation algorithm, where Calc_U() calculates all expansion coefficients $u^{\eta}_{\mu,\mu'}$ for an atom $i$ while Calc_Z() calculates the partial sums $Z^{\mu,\mu'}_{\eta_1,\eta_2,\eta}$, which are defined as

$$Z^{m,m'}_{j_1,j_2,j} = \sum_{m_1,m'_1=-j_1}^{j_1} \sum_{m_2,m'_2=-j_2}^{j_2} H^{j,m,m'}_{j_1,m_1,m'_1,j_2,m_2,m'_2} u^{j_1}_{m_1,m'_1} u^{j_2}_{m_2,m'_2}. \tag{7.1}$$

In the loop over neighbors, first the derivatives of $\mathbf{u}_i$ with respect to the distance vector be-

**Compute_SNAP():**
for $i$ in natoms() {
  $\mathbf{u}_i$ = Calc_U($i$)
  $\mathbf{Z}_i$ = Calc_Z($i$,$\mathbf{u}_i$)
  for $j$ in neighbors($i$) {
    $\nabla_j \mathbf{u}_i$ = Calc_dUdR($i$, $j$,$\mathbf{u}_i$)
    $\nabla_j \mathbf{B}^i$ = Calc_dBdR($i$, $j$,$\mathbf{u}_i$,$\mathbf{Z}_i$,$\nabla_j \mathbf{u}_i$)
    $\mathbf{F}_{ij} = -\boldsymbol{\beta} \cdot \nabla_j \mathbf{B}^i$
    $\mathbf{F}_i$ += $-\mathbf{F}_{ij}$; $\mathbf{F}_j$ += $\mathbf{F}_{ij}$
} }

**Figure 7.1.** Base algorithm for the SNAP force calculation.

**Function Calc_dBdR($i$, $j$):**
for $(\eta, \eta_1, \eta_2)$ in GetBispectrumIndices() {
  $\nabla_j B_{\eta_1,\eta_2,\eta} = 0$
  for $(\mu = 0; \mu \leq \eta; \mu$++ ) {
    for $(\mu' = 0; \mu' \leq \eta; \mu'$++) {
      $\nabla_j Z^{\mu,\mu'}_{\eta_1,\eta_2,\eta} = 0$
      for $(\mu_1 = \max(0, \mu + (\eta_1 - \eta_2 - \eta)/2);$
          $\mu_1 \leq \min(\eta_1, \mu + (\eta_1 + \eta_2 - \eta)/2); \mu_1$++) {
        for $(\mu'_1 = \max(0, \mu' + (\eta_1 - \eta_2 - \eta)/2);$
            $\mu'_1 \leq \min(\eta_1, \mu' + (\eta_1 + \eta_2 - \eta)/2); \mu'_1$++) {
          $\mu_2 = \mu - \mu_1$ ; $\mu'_2 = \mu' - \mu'_1$
          $\nabla_j Z^{\mu,\mu'}_{\eta_1,\eta_2,\eta}$ += $H^{\eta,\mu,\mu'}_{\eta_1,\mu_1,\mu'_1,\eta_2,\mu_2,\mu'_2}$
            $(u^{\eta_1}_{\mu_1,\mu'_1} \nabla_j u^{\eta_2}_{\mu_2,\mu'_2} + u^{\eta_2}_{\mu_2,\mu'_2} \nabla_j u^{\eta_1}_{\mu_1,\mu'_1})$
      } }
      $\nabla_j B_{\eta_1,\eta_2,\eta}$ += $(u^{\eta}_{\mu,\mu'})^* \nabla_j Z^{\mu,\mu'}_{\eta_1,\eta_2,\eta} + Z^{\mu,\mu'}_{\eta_1,\eta_2,\eta} (\nabla_j u^{\eta}_{\mu,\mu'})^*$
} } }

**Figure 7.2.** Original algorithm for the derivative of the bispectrum components of atom $i$ w.r.t. the position of atom $j$ using Eq. 7.2.

tween atoms $i$ and $j$ are computed in Calc_dUdR() and then the derivatives of $\mathbf{B}^i$ are computed in Calc_dBdR(), which is the most computationally expensive part of the algorithm. For the parameter sets used in this study, Calc_dBdR() is responsible for approximately 90% of all floating point and memory operations. Thus, we concentrate our description on this function. In the original calculation, the spatial derivative of the bispectrum components was written as

$$\begin{aligned}
\nabla B_{j_1,j_2,j} &= \sum_{m,m'}^{j} (\nabla u^{j}_{m,m'})^* \sum_{m_1,m_1'}^{j_1} H^{\substack{jmm' \\ j_1 m_1 m_1' \\ j_2 m_2 m_2'}} u^{j_1}_{m_1,m_1'} u^{j_2}_{m_2,m_2'} \\
&+ \sum_{m,m'}^{j} (u^{j}_{m,m'})^* \sum_{m_1,m_1'}^{j_1} H^{\substack{jmm' \\ j_1 m_1 m_1' \\ j_2 m_2 m_2'}} \nabla u^{j_1}_{m_1,m_1'} u^{j_2}_{m_2,m_2'} \\
&+ \sum_{m,m'}^{j} (u^{j}_{m,m'})^* \sum_{m_1,m_1'}^{j_1} H^{\substack{jmm' \\ j_1 m_1 m_1' \\ j_2 m_2 m_2'}} u^{j_1}_{m_1,m_1'} \nabla u^{j_2}_{m_2,m_2'}
\end{aligned} \tag{7.2}$$

where the symbol $\nabla$ denotes the derivative of what follows with respect to the position of some neighbor atom. We have dropped the double summation over $m_2$ and $m_2'$, as the coupling coefficients are non-zero only for $m_2 = m - m_1$, likewise for $m_2'$. In the first term, the inner double sum over $m_1$ and $m_1'$ contains no derivatives, and so can be pre-calculated. Hence, for the highest order bispectrum component ($j_1 = j_2 = j = J$), the computational complexity of this term is $O(J^2)$. However, in the second and third term, the inner double sums contain derivatives and so must be calculated separately for each neighbor atom and for each entry in the outer double sums over $m$ and $m'$. As a result, the computation complexity of the second and third terms is $O(J^4)$. In Fig. 7.2 we show the algorithm for Calc_dBdR() based on Eq. 7.2.

To examine the parallel performance of the SNAP potential, we ran studies on three high performance computing platforms [36]: Chama at SNL (Intel Sandybridge cluster), Sequoia/Vulcan at LLNL (IBM BGQ), and Titan at ORNL (Nvidia GPU). The relevant technical attributes of these machines are summarized in Table 7.1 In all cases, parallelism over nodes was achieved using the standard LAMMPS spatial decomposition framework [19]. On Chama and Sequoia, additional parallelism within each node was achieved by distributing the calculations associated with each neighbor atom over the available cores on the node [36]. On Titan, even greater parallelism was achieved by decomposing the force contribution due to a single neighbor over many threads. For all of the tests, micro load balancing was used to achieve good parallel efficiency when the number of nodes approaches the number of atoms.

Figure 7.3 shows the strong scaling performance of a 246K atom system on the three HPC platforms. All platforms show nearly ideal scaling over most of the range. At the same node count, the absolute time to solution on Chama and Titan was about the same, while Sequoia took about 5x longer. In the case of Sequoia, we were able to scale the fixed size problem all the way from a single node to the entire machine (122,880 nodes). At this point, there were only 2 atoms per node, the time to solution was still decreasing, and the parallel efficiency was 14%.

| System | Nodes | Cores | Threads | Power/Node | Perf./Node |
|---|---|---|---|---|---|
| Chama | 1,230 | 19,680 | 39,360 | 368.8 W | 332.8 GFLOP/s |
| Sequoia/Vulcan | 122,880 | $2.0 \times 10^6$ | $7.9 \times 10^6$ | 80.26 W | 204.8 GFLOP/s |
| Titan | 18,688 | $5.0 \times 10^7$ | $5.4 \times 10^8$ | 439.3 W | 1450.8 GFLOP/s |

**Table 7.1.** Relevant hardware parameters of the platforms used for strong scaling experiments. Data was obtained from the November 2013 Top500 list [1].

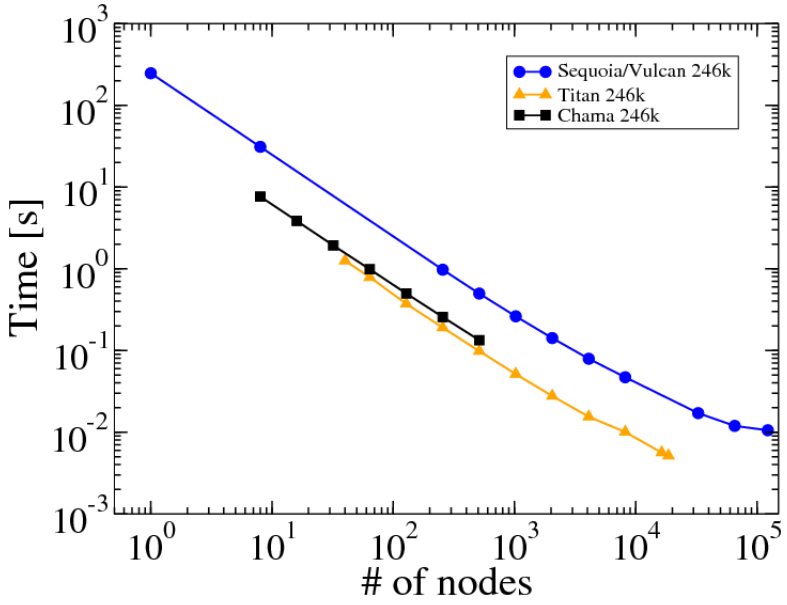


**Figure 7.3.** Strong scaling performance of SNAP on three HPC platforms.

When scaling out to full system size Titan is about two times faster than Sequoia/Vulcan. Most of this factor is due to Titan having more atoms per node at full scale ($\sim$ 13 versus 2 on Sequoia/Vulcan). Thus the surface to volume ratio from the domain decomposition is better and the GPUs are operating in a range where they can still be filled effectively. This is reflected by the parallel efficiencies which drop only to about 50% on Titan compared with 14% on Sequoia/Vulcan. Chama is not large enough to show any significant loss of parallel efficiency for a system of 246k atoms. Normalizing the node count by power consumption per node in Fig. 7.4 makes the relative energy efficiency much clearer. For small to medium node counts, the times to solution are within 20% for all three systems, at the same total power. In this regime Sequoia/Vulcan is actually the most effective one followed by Titan and then Chama. Only at larger node counts (i.e. less work per node) does Titan become more efficient than Sequoia/Vulcan. The crossover point is reached at about 200 atoms per GPU.

In addition to running large numbers of atoms, we spent a significant effort improving the

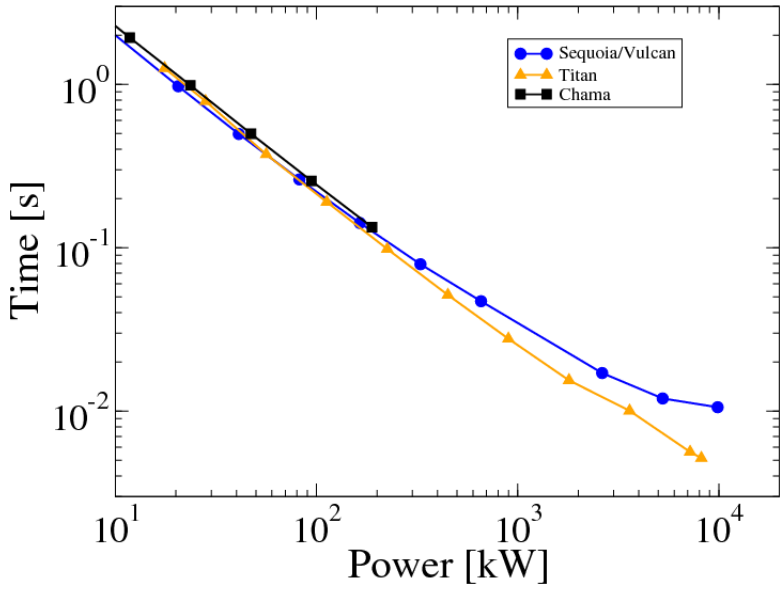


**Figure 7.4.** Power consumption of the three HPCs with the SNAP algorithm.

way that the bispectrum components are calculated. The non-trivial permutation symmetry in the bispectrum indices given by Eq. 2.7 allows us to rewrite Eq. 7.2 as

$$\begin{aligned}\nabla B_{j_1,j_2,j} \;=\; & \sum_{m,m'}^{j} (\nabla u^{j}_{m,m'})^* \sum_{m_1,m_1'}^{j_1} H^{\,jmm'}_{j_1 m_1 m_1'\, j_2 m_2 m_2'}\; u^{j_1}_{m_1,m_1'} u^{j_2}_{m_2,m_2'} \\ & + \frac{2j+1}{2j_1+1} \sum_{m_1,m_1'}^{j_1} (\nabla u^{j_1}_{m_1,m_1'})^* \sum_{m,m'}^{j} H^{\,j_1 m_1 m_1'}_{j mm'\, j_2 m_2 m_2'}\; u^{j}_{m,m'} u^{j_2}_{m_2,m_2'} \\ & + \frac{2j+1}{2j_2+1} \sum_{m_2,m_2'}^{j_2} (\nabla u^{j_2}_{m_2,m_2'})^* \sum_{m_1,m_1'}^{j_1} H^{\,j_2 m_2 m_2'}_{j_1 m_1 m_1'\, jmm'}\; u^{j_1}_{m_1,m_1'} u^{j}_{m,m'}\end{aligned} \tag{7.3}$$

Written in this way, all of the inner double sums are free of derivatives and can be pre-calculated. This has the effect of reducing overall the computational complexity from $O(J^4)$ to $O(J^2)$. In Fig. 7.5 we show the improved algorithm based on Eq. 7.3 that takes advantage of the symmetry relation Eq. 2.7.

In Fig. 7.6 we compare the performance of the original and improved algorithms. Timings are based on a 10,000 step MD simulations of BCC tantalum crystal using the SNAP potential described in Section 4.1. The calculations were performed on Sandia's Chama high-performance cluster with a dual socket Intel Sandy Bridge processor with 16 cores on each node. Three different system sizes were used, containing 512, 4096, and 32768 atoms. Each system size was run on 8,

**Function Calc_dBdR($i$, $j$):**

for $(\eta, \eta_1, \eta_2)$ in GetBispectrumIndices() {
  $\nabla_j B_{\eta_1,\eta_2,\eta} = 0$
  for $(\mu = 0; \mu \leq \eta; \mu$++ ) {
    for $(\mu' = 0; \mu' \leq \eta; \mu'$++) {
      $\nabla_j B_{\eta_1,\eta_2,\eta}$ += $Z^{\mu,\mu'}_{\eta_1,\eta_2,\eta} (\nabla_j u^{\eta}_{\mu,\mu'})^*$
  } }
  for $(\mu_1 = 0; \mu_1 \leq \eta_1; \mu_1$++ ) {
    for $(\mu'_1 = 0; \mu'_1 \leq \eta_1; \mu'_1$++) {
      $\nabla_j B_{\eta_1,\eta_2,\eta}$ += $\frac{\eta+1}{\eta_1+1} Z^{\mu_1,\mu'_1}_{\eta,\eta_2,\eta_1} (\nabla_j u^{\eta_1}_{\mu_1,\mu'_1})^*$
  } }
  for $(\mu_2 = 0; \mu_2 \leq \eta_2; \mu_2$++ ) {
    for $(\mu'_2 = 0; \mu'_2 \leq \eta_2; \mu'_2$++) {
      $\nabla_j B_{\eta_1,\eta_2,\eta}$ += $\frac{\eta+1}{\eta_2+1} Z^{\mu_2,\mu'_2}_{\eta_1,\eta,\eta_2} (\nabla_j u^{\eta_2}_{\mu_2,\mu'_2})^*$
  } }
}

**Figure 7.5.** Improved algorithm for the derivative of the bispectrum components of atom $i$ w.r.t. the position of atom $j$ using Eq. 7.3.

16, 32, 64, 128, and 256 nodes. We plot the time required to calculate one MD time step versus the number of atoms per node. In this form, results for the three different system sizes are almost indistinguishable, indicating that both single node performance and strong scaling efficiency are determined primarily by the number of atoms per node. When the number of atoms per node is large, the parallel scaling is close to ideal, and the improved algorithm is consistently about 16x faster than the original algorithm. As the number of atoms per node decreases, this margin also decreases, due to the parallel efficiency of the improved algorithm decreasing more rapidly.

Finally, Figure 7.7 shows where SNAP fits on the computational cost "map" of interatomic potentials. The map was generated by plotting the LAMMPS time per timestep for each potential versus the year in which the potential was published. Generally, the computational cost of potentials is increasing exponentially over time. The computational cost of the original SNAP algorithm is consistent with this trend. However, the larger speed-up achieved with the improved algorithm has allowed us to buck this trend, and SNAP is now on a par with other high-accuracy potentials such as ReaxFF and COMB.

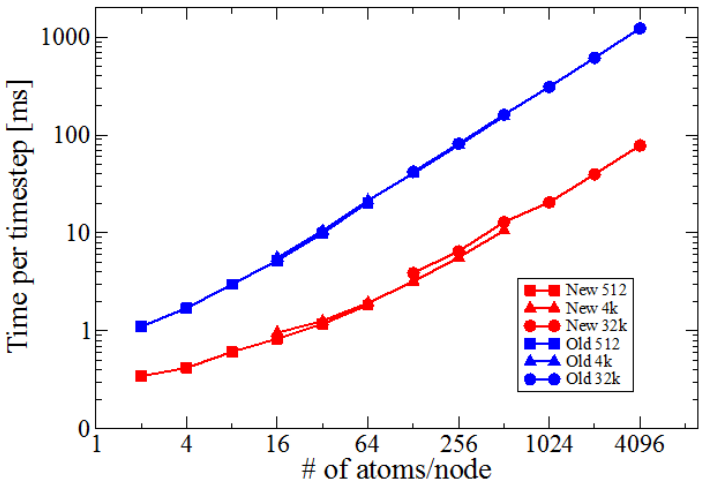


**Figure 7.6.** CPU time per MD time step versus atoms per node for benchmark simulations of BCC tantalum using the original and improved implementations of the SNAP potential. Results are shown for systems containing 512, 4096, and 32768 atoms.

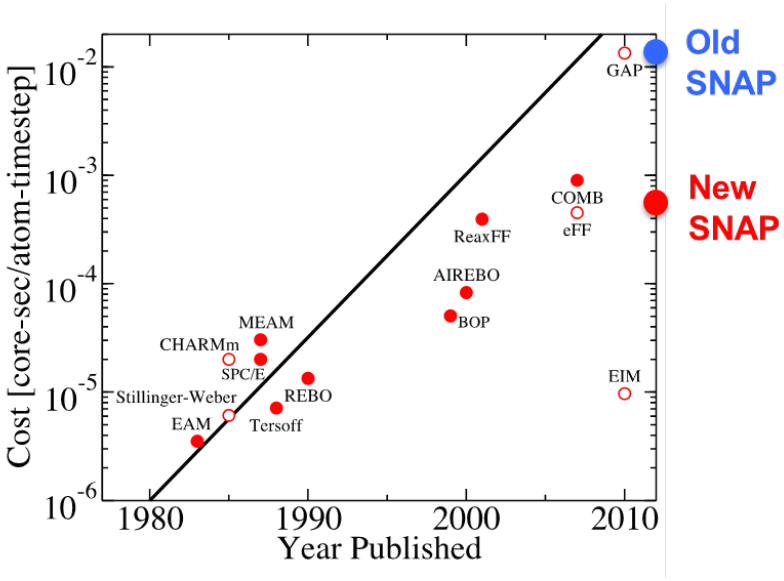


**Figure 7.7.** The computational cost of the original and improved SNAP implementations relative to other interatomic potentials in LAMMPS.

# Chapter 8

# Electrostatics, MSM method

Since simulations of atomistic, molecular, and coarse-grained systems commonly include point-charge models, they require the calculation of Coulombic 1/r interactions. And since Coulombic interactions are inherently long-range in nature, molecular simulations with periodic boundary conditions that include them must be handled in a special way. Common methods for dealing with long-range electrostatic interactions include the Ewald sum method, and the FFT-based particle-particle/particle-mesh (PPPM) method. Unfortunately, the Ewald sum scales poorly and does not perform well for large systems, and the PPPM method requires FFTs, which does not scale very well for large core counts. The multilevel summation method (MSM) is a newer alternative that does not require FFTs and promises to scale better than those older methods.

We have written a parallel version of MSM using domain decomposition for LAMMPS [19], and have evaluated its performance against LAMMPS's implementation of Ewald and PPPM for several typical MD simulation problems. We have also enhanced the MSM method and made all of these developments freely available in LAMMPS [19], and have published our findings [26].

In both Ewald-based methods and MSM, the Coulombic interaction is split into a short-range part and a long-range part, as in Figure 8.1, pane (a). Pane (b) of Figure 8.1 shows that for the MSM algorithm, the long-range part is further split into several different levels. Figure 8.2 provides further pictorial elaboration of the MSM algorithm concept. One of the improvements we made to the MSM method involves using a half-sphere of interaction for the MSM direct part as shown in Figure 8.3.

To test the performance of our MSM implementation in LAMMPS, we ran several test problems and compared against LAMMPS's Ewald and PPPM implementations. Figure 8.4 shows the results of one of our strong scaling tests (ranging from 1 to 512 cores) on the Redsky platform for the SPC/E water benchmark, using an estimated relative RMS force accuracy of 0.001. These results represent the total run time and long-range (i.e., kspace) time as reported in the LAMMPS log file. For total run time on a single processor, PPPM is 1.6 times faster than MSM, while Ewald is 1.9 times slower than MSM as shown in Figure 8.4(a). For long-range time on a single processor, PPPM is 10.9 times faster than MSM, and Ewald is 3.0 times slower than MSM as shown in Figure 8.4(b). However, for total run time on 512 cores, MSM is 2.3 times faster than PPPM running on 64 cores, and 1.5 times faster than Ewald running on 512 cores.

All told, we have added several new extensions and improvements to MSM. We implemented and tested a method for computing the full pressure tensor using MSM and found that computing

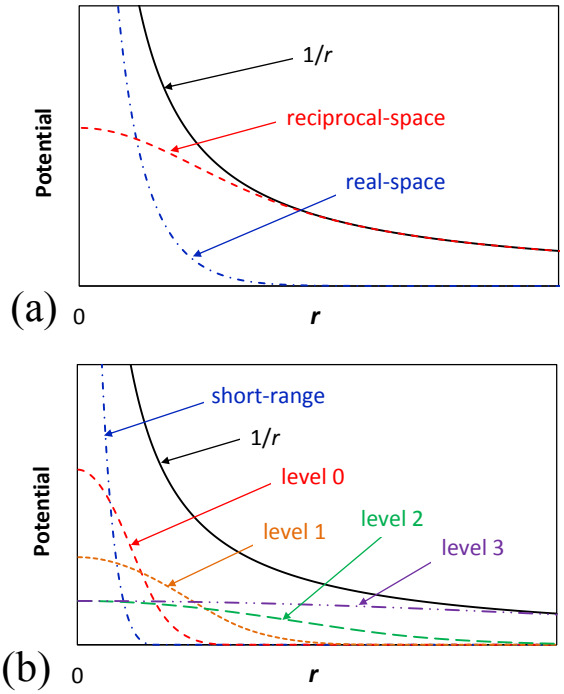


**Figure 8.1.** Conceptual splitting of the 1/r potential for (a) Ewald-based methods and (b) MSM with four levels.

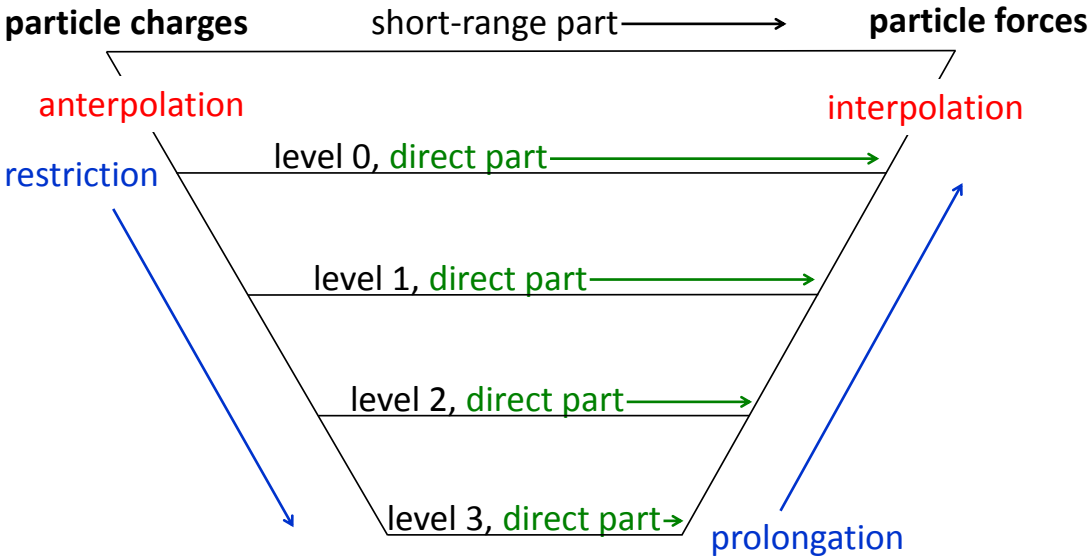


**Figure 8.2.** Algorithmic steps for MSM with four grid levels.

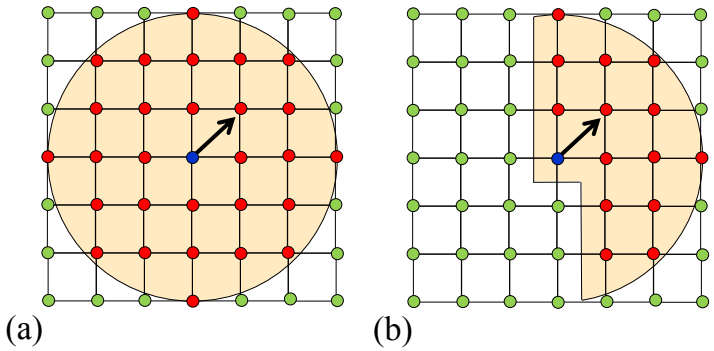


**Figure 8.3.** Conceptual diagram (in 2D) of (a) the original sphere of interaction for the direct part and (b) the half-sphere. In practice, interactions between the central grid point and all grid points inside the dashed region are computed.

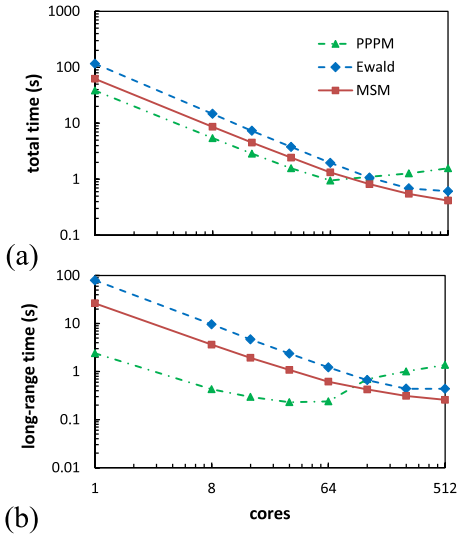


**Figure 8.4.** CPU time (s) of running the SPC/E water system on Redsky using 0.001 relative accuracy for (a) total run time and (b) long-range portion.

the pressure tensor using MSM is relatively expensive as compared to computing the pressure tensor using other long-range electrostatics methods such as PPPM. A faster way to calculate the scalar pressure has also been tested. In order to make fair comparisons between the various methods for computing electrostatic interactions, accurate error estimation methods are needed, so we improved the usability of the MSM error estimation algorithm, making it compatible with the error estimation methods for the other electrostatics methods in LAMMPS (PPPM and Ewald). In addition, we extended MSM to non-orthogonal (triclinic) systems in a manner that requires only a

few changes to the original algorithm. On a single processor, the direct part in MSM is typically the bottleneck, and the performance of MSM has been significantly improved by using a half-sphere for direct part interactions instead of a full sphere.

On a single processor, we found our implementation of MSM to be faster than the implementation of MSM in the NAMD-lite code. However, despite significant improvements to MSM, PPPM was still found to be faster for both a SPC/E water system and a solvated rhodopsin protein system running on a single processor. The Ewald sum was also found to be slower than PPPM on a single processor, especially for large systems.

In summary, we have written a parallel version of MSM using the highly scalable domain decomposition method on a distributed memory architecture. Since FFTs represent a major scaling bottleneck for the PPPM method when running on many cores, it is hoped that a non-FFT method like MSM can help alleviate this issue. For both the water and the rhodopsin protein test problems running on Sandia's Redsky machine, MSM is faster than PPPM when using many cores (when a many-to-many communication pattern is expensive). However, using only a subset of processors for PPPM can help to reduce the FFT scaling bottleneck. MSM is more competitive for relatively low accuracy. On Sandia's Chama machine however, PPPM was found to perform better than MSM for all core counts that we tested. This suggests that PPPM should be used for problem sizes and core counts typically run using current high performance computers. However, the MSM algorithm is still relatively new as compared to well-established FFT-based methods such as PPPM, and we anticipate that further improvements to the MSM algorithm could enhance its competitiveness for calculation of long-range electrostatic interactions. MSM also allows fully non-periodic long-range electrostatics calculations, which are not possible using Ewald-based methods, and in a more efficient manner than the fast multipole method (FMM).

We have made our implementation of the MSM algorithm and enhancements freely available in the LAMMPS [19] molecular simulation package. For more details, please see our full report [26].

# Chapter 9

# Summary

The original goal of this project was to develop a capability for on-demand, automated generation of new potentials for arbitrary materials. We have achieved this goal through the creation of the SNAP (Spectral Neighbor Analysis Potential) potential, which is embodied in the SNAP standard LAMMPS package and the FitSnap.py fitting software. The SNAP potential has a very general form and uses machine-learning techniques to reproduce the energies, forces, and stress tensors of a large set of small configurations of atoms, which are obtained using high-accuracy quantum electronic structure (QM) calculations. The local environment of each atom is characterized by a set of bispectrum components of the local neighbor density projected on to a basis of hyperspherical harmonics in four dimensions. The SNAP coefficients are determined using weighted least-squares linear regression against the full QM training set. This allows the SNAP potential to be fit in a robust, automated manner to large QM data sets using many bispectrum components. The calculation of the bispectrum components and the SNAP potential are implemented in the LAMMPS parallel molecular dynamics code. The FitSnap.py software is a robust fitting package that calculates the least-squares SNAP coefficients for an arbitrary QM dataset.

## 9.1 Significant Research accomplishments

The significant research accomplishments of this LDRD project are:

- Development of the SNAP potential.
- Generation of SNAP potentials for selected materials (Ta, InP, Si/SiO2).
- Scalable, efficient implementation of SNAP within LAMMPS.
- Robust fitting package (FitSnap.py) that interfaces to LAMMPS and DAKOTA to determine optimal SNAP coefficients.
- Five external publications and seven invited talks.
- Two technical advance disclosures filed.

The two technical advance disclosures that were filed as part of this LDRD are:

- SD no. 12361: Linear Spectral Neighbor Analysis for Interatomic Potential Energy (2012). Submitted by Aidan Thompson and Laura Swiler.

- SD no. 13251: Efficient Algorithm for Derivatives of SNAP Bispectrum Components (2014). Submitted by Aidan Thompson, Laura Swiler, Christian Trott, Stan Moore, and Jonathan Moussa.

The publications that resulted from this LDRD are:

- S. J. Plimpton and A. P. Thompson , "Computational aspects of many-body potentials," *Mat. Res. Soc. Bulletin* **37** 513 (2012).

- S. Levy, K. B. Ferreira, A. P. Thompson , and C. Trott, "Evaluating the Feasibility of Using Memory Content Similarity," *Inter. J. of High Perf. Comp.* (2013).

- S.G. Moore and P.S. Crozier, "Extension and evaluation of the multilevel summation," *J. Chem. Phys.* **140** 234112 (2014).

- C. R. Trott, S. D. Hammond, A. P. Thompson, "SNAP: Strong scaling high fidelity molecular dynamics simulations," *Lecture Notes in Comp. Sci.* 7905 (2014).

- A.P. Thompson, L.P. Swiler, C.R. Trott, S.M. Foiles, G. Tucker, "A Machine-Learning Method for Automated Generation of Quantum-Accurate Interatomic Potentials," *J. Comp. Phys.* (submitted)(2014).

The external presentations that resulted from this LDRD are:

- CIS External Review 2013, Poster Presentation, Quantum-Accurate LAMMPS SNAP: Simulations on Petascale Platforms, A.P. Thompson, C.R. Trott, L.P. Swiler, G. Tucker, S.M. Foiles, and S.J. Plimpton.

- A.P. Thompson, Materials Research Society Annual Meeting, San Francisco, CA, April 2014

- A.P. Thompson, American Chemical Society Annual Meeting, Indianapolis, IN, September 2013

- A.P. Thompson, Materials Design Inc., San Francisco, CA, October 2012

- A.P. Thompson, ExxonMobil Research, Annandale, NJ, April 2013

- S.M. Foiles, A. P. Thompson, L. P. Swiler, G. Tucker, C. R. Trott, C. Weinberger, TMS Annual Meeting 2014, San Diego, CA, February 2014.

- C.R. Trott, S.D. Hammond, A.P. Thompson, Inter. Supercomp. Conf., Leipzig, Germany, July 2014.

Overall, the most significant accomplishment is the development of the SNAP potential and its implementation in LAMMPS. The FitSnap.py software for generating new SNAP potentials is available to Sandians, including training data, examples, and the potentials developed thus far. The SNAP tantalum potential is currently being used in the Predictive Performance Margins (PPM) project.

## 9.2 Future work

We anticipate developing SNAP potentials for other materials. We also are working with various groups on data-driven potentials. We will continue to improve the SNAP code in LAMMPS. Finally, we plan to perform large-scale LAMMPS simulations with SNAP potentials on Sequoia and other advanced platforms.

# References


[1] TOP500 Supercomputer Site, `http://www.top500.org`.

[2] B. M. Adams, K. R. Dalbey, M. S. Eldred, D. M. Gay, L. P. Swiler, W. J. Bohnhoff, J. P. Eddy, and K. Haskell. Dakota users manual version 5.1. Sandia Technical Report SAND2010-2183, Sandia National Laboratories, Albuquerque, NM, 2011.

[3] N. Artrith and J. Behler. High-dimensional neural network potentials for metal surfaces: A prototype study for copper. *Physical Review B*, 85:045439, 2012.

[4] A.P. Bartok, M. J.Gillan, F. R. Manby, and G. Csanyi. On representing chemical environments. *Physical Review B*, 87:184115, 2013.

[5] A.P. Bartok, M. C. Payne, K. Risi, and G. Csanyi. Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons. *Physical Review Letters*, 104:136403, 2010.

[6] Albert Bartok-Partay. *The Gaussian Approximation Potential: An Interatomic Potential Derived from First Principles Quantum Mechanics*. Springer Theses, 2010.

[7] F. H. Featherston and J.R. Neighbours. Elastic Constants of Tantalum, Tungsten and Molybdenum. *Physical Review*, 130():1324–1333, 1963.

[8] J. N. Glosli, D. F. Richards, K. J. Caspersen, R. E. Rudd, J. A. Gunnels, and F. H. Streitz. Extending Stability Beyond CPU Millennium: A Micron-scale Atomistic Simulation of Kelvin-Helmholtz Instability. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, SC '07, pages 58:1–58:11, New York, NY, USA, 2007. ACM.

[9] G. Goedecker. Linear Scaling Electronic Structure Methods. *Rev. Mod. Phys*, 71:1085, 1999.

[10] M. Griebel, S. Knapek, and G. Zumbusch. *Numerical Simulation in Molecular Dynamics*. Springer Verlag, Heidelberg Germany, 2007.

[11] R. Gröger, A. G. Bailey, and V. Vitek. Multiscale modeling of plastic deformation of molybdenum and tungsten: Atomistic studies of the core structure and glide of $1/2< 111 >$ screw dislocations at $0^{\circ}$K. *Acta Materialia*, 56:5401–5411, 2008.

[12] T. Hamada, T. Narumi, R. Yokota, K. Yasuoka, K. Nitadori, and M. Taiji. 42 TFlops Hierarchical N-body Simulations on GPUs with Applications in Both Astrophysics and Turbulence. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 62:1–62:12, New York, NY, USA, 2009. ACM.

[13] K. Kadau, T.C. Germann, and P.S. Lomdahl. Molecular Dynamics Comes of Age: 320 Billion Atom Simulation on BlueGene/L. *International Journal of Modern Physics C*, 17(12):1755–1761, 2006.

[14] K. W. Katahara, M. H. Manghnani, and E. S. Fisher. Pressure derivatives of the elastic moduli of niobium and tantalum. *Journal of Applied Physics*, 47():434–439, 1976.

[15] M. Kim, K. H. Khoo, and J. R. Chelikowsky. Simulating liquid and amorphous silicon dioxide using real-space pseudopotentials. *Phys. Rev. B*, 86:054104, 2012.

[16] G. Kresse and J. Furthmller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 54(16):11169–11186, 1996.

[17] G. Kresse and D. Joubert. From ultrasoft pseudopotentials to the projector augmented-wave method. *Physical Review B*, 59():1758–1775, 1999.

[18] L. D. Landau and E. M. Lifshitz. *Course of Theoretical Physics, Volume 5 Statistical Physics, Part 1*. Elsevier Butterworth Heinemann, Oxford, 1980.

[19] LAMMPS. Lammps molecular dynamics package www site: lammps.sandia.gov. potential benchmarks: lammps.sandia.gov/bench.html.

[20] Y. Li, D. J. Siegel, J. B. Adams, and X.-Y. Liu. Embedded-atom-method tantalum potential developed by the force-matching method. *Physical Review B*, 67:125101, 2003.

[21] A. E. Mattsson, P. A. Schultz, M. P. Desjarlais, T. R. Mattsson, and K. Leung. Designing meaningful density functional theory calculations in material science–A primer. *Modelling Simul. Mater. Sci. Eng.*, 13:R1–R31, 2005.

[22] A. V. Meremianin. Multipole expansions in four-dimensional hyperspherical harmonics. *Journal Of Physics A-Mathematical And General*, 39(12):3099–3112, MAR 24 2006.

[23] A. V. Meremianin. Hyperspherical harmonics with arbitrary arguments. *Journal Of Mathematical Physics*, 50(1), JAN 2009.

[24] Y. Mishin, A.Y. Lozovoi, and H.N.G. Wadley. Angular-dependent interatomic potential for tantalum. *Acta Materialia*, 54(19):5013–5026, 2006.

[25] H. J. Monkhorst and J.D. Pack. Special points for Brillouin-zone integrations. *Physical Review B*, 13(12):5188–5192, 1976.

[26] S. G. Moore and P. S. Crozier. Extension and evaluation of the multilevel summation method for fast long-range electrostatics calculations. *Journal of Chemical Physics*, 140:234112, 2014.

[27] D.A. Murdick, X.W. Zhou, H.W.G. Wadley, D. Nguyen-Manh, R. Drautz, and D.G. Pettifor. Analytic bond-order potential for the gallium arsenide system. *Phys. Rev. B*, 73:045206, 2006.

[28] J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized Gradient Approximation Made Simple. *Physical Review Letters*, 77(18):3865–3868., 1996.

[29] J.P. Perdew and A. Zunger. Self-interaction correction to density-functional approximations for many-electron systems. *Phys. Rev. B*, 23:5048, 1981.

[30] S Plimpton. Fast Parallel Algorithms For Sort-Range Molecular-Dynamics. *J. Comput. Phys*, 117(1):1–19, 1995.

[31] P.A. Schultz. Simple intrinsic defects in inp: Numerical predictions. Sandia Technical Report SAND2012-3313, Sandia National Laboratories, Albuquerque, NM, 2013.

[32] P. J. Steinhardt, D. R. Nelson, and M. Ronchetti. Bond-Orientational Order In Liquids And Glasses. *Physical Review B*, 28(2):784–805, 1983.

[33] G. Strang. *Linear Algebra and Its Applications*. Academic Press, New York, NY, 1980.

[34] S. Swaminarayan, T.C. Germann, K. Kadau, and G.C. Fossum. 369 Tflop/s Molecular Dynamics Simulations on the Roadrunner General-Purpose Heterogeneous Supercomputer. pages 1–10, Nov 2008.

[35] A.P. Thompson, L.P. Swiler, C.R. Trott, S.M. Foiles, and G. Tucker. SNAP: Automated Generation of Quantum-Accurate Interatomic Potentials. *J. Comp. Physics*, 2014.

[36] C. R. Trott, S. D. Hammond, and A. P. Thompson. SNAP: Strong scaling high fidelity molecular dynamics simulations on leadership-class computing platforms. In *Proceedings of the 30th International Supercomputing Conference, ISC 2013, Leipzig, Germany, June 22-26, 2014.*, volume 7905 of *Lecture Notes in Computer Science*. Springer Verlag, 2014.

[37] D. A. Varshalovich, A. N. Moskalev, and V. K. Khersonskii. *Quantum Theory of Angular Momentum*. World Scientific, Singapore, 1987.

[38] Lin-Wang Wang, Byounghak Lee, Hongzhang Shan, Zhengji Zhao, Juan Meza, Erich Strohmaier, and David H. Bailey. Linearly scaling 3d fragment method for large-scale electronic structure calculations. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08, pages 65:1–65:10, Piscataway, NJ, USA, 2008. IEEE Press.

[39] C.R. Weinberger, G.J. Tucker, and S.M. Foiles. Peierls potential of screw dislocations in bcc transition metals: Predictions from density functional theory. *Phys. Rev. B*, 87:054114, 2013.

[40] X. W. Zhou, R. A. Johnson, and H. N. G. Wadley. Misfit-energy-increasing dislocations in vapor-deposited cofe/nife multilayers. *Phys. Rev. B*, 69:144113, Apr 2004.

[41] J.F. Ziegler, J. P. Biersack, and U. Littmark. The Stopping and Range of Ions in Matter. 1, 1985.

# Appendix A

# fitsnap.py User's Manual

## A.1 Introduction

Fitsnap.py, together with LAMMPS, determines the optimal set of SNAP coefficients (in a least-squares sense; see Equation 2.16) provided a training set of atomic configurations with energies, forces, and virial components. It is written entirely in Python and will run on any platform where LAMMPS can run.

The basic steps of its operation are:

1. Read in JSON-format training set data and hyperparameters and write out LAMMPS data files.
2. Run LAMMPS to calculate
    - Bispectrum components and their contributions to the forces and virials of the training set configurations.
    - Reference potentials (e.g. coulombic interactions, ZBL) for the training set configurations.
3. Read in LAMMPS output and construct the linear system in Equation 2.14.
4. Solve the system and write out the resulting SNAP potential and regression residuals.

The fitting process is controlled using two input files. A master input file containing a listing of keyword/value pairs controls operation of fitsnap.py and also sets the hyperparameters that determine how LAMMPS generates bispectrum components. The remaining hyperparameters, the regression weights, are communicated in a separate file, grouplist.in. Complete descriptions of the keywords and expected format of these two files are provided in Sections A.4 and A.6.

By caching reusable data—the parsed and converted JSON-format training set, for example—fitsnap.py reduces the time required to generate SNAP potentials, which enables fuller exploration of hyperparameter space in the context of Dakota-driven sensitivity analyses or optimizations. Additionally, fitsnap.py can take full advantage of multicore workstations and clusters by invoking LAMMPS in parallel.

Fitsnap.py requires a minimum of Python 2.4. It is untested with Python 3.x. It also requires SciPy and NumPy and a LAMMPS executable that has been built with the SNAP standard LAMMPS package. Parallel operation requires LAMMPS to be built with MPI support.

## A.2 Command Line Options

The command line options accepted by fitsnap.py are shown below. If no options are given, fitsnap.py listens on standard input for piped or redirected input.

`-h, --help`

Show a help message and exit.

`-c <element names>, --convert-JSON <element names>`

Only convert JSON-format training set data for later (re)use. (Do not run LAMMPS or perform a fit.) User must supply a label for each chemical element in the training data.

`-i <file>, --input=<file>`

Use the provided master input file.

## A.3 Examples

Create a SNAP potential using the input file InP.in.

```
fitsnap.py < InP.in
```

Equivalent to the previous example.

```
fitsnap.py --input=InP.in
```

Convert the JSON-format training set configurations into LAMMPS data files and to a serialized format. On subsequent runs, fitsnap.py will detect and use the serialized data, which is considerably faster than re-converting the JSON-format files.

```
fitsnap.py --convert-JSON In P
```

## A.4 Master Input File

The master input file is used to control the operation of fitsnap.py and to provide some of the hyperparameters that affect generation of the SNAP potential.

The format of the input file is simple: one keyword per line, followed by an equal sign (=), followed by the value to assign to the keyword. Arbitrary whitespace is permitted with the caveat that the keyword, equal sign, and value must be on the same line. Text following a pound sign (#) will be ignored to the end of the line, and comments are permitted on the same lines as keywords.

Strings, integers, and floating point numbers are valid 'types' of keyword arguments, with integers indicating in a few cases a binary on(1)/off(0) state. It is not necessary to quote strings unless they contain whitespace. Most input is subjected to some validation; for example, only values [0..3] are permitted for diagonalstyle, and an error will result if any other values are used.

The following is an alphabetized listing of all the keywords understood by fitsnap.py, including whether they are required or optional, their types, limits or restrictions on their values, default values, and a brief explanation of their intended usage.

### diagonalstyle

*Optional*
*Type:* Integer
*Range:* 0, 1, 2, or 3
*Default:* 3
*Description:* Along with the keyword twojmax, determines which bispectrum components are included in the expansion. See Section 2.1 for definitions.

0: all $j_1$, $j_2$, $j <=$ twojmax, $j_2 <= j_1$
1: subset satisfying $j_1 == j_2$
2: subset satisfying $j_1 == j_2 == j_3$
3: subset satisfying $j_2 <= j_1 <= j$

### groupFile

*Optional*
*Type:* String
*Range:* None
*Default:* grouplist.in
*Description:* Name of the file that contains the weighting for the training set groups. See Group Weights for file format and further explanation.

### lammpsPath

*Optional*
*Type:* String
*Range:* None
*Default:* ./lmp.exe

*Description:* Path to the LAMMPS executable. The path can be absolute or relative to where fitsnap.py is run.

## maxConcurrency

*Optional*
*Type:* Integer
*Range:* >=0
*Default:* 1
*Description:* Maximum number of configurations that LAMMPS should evaluate concurrently.
0: Run LAMMPS using the number of cores detected on this machine
1: Run LAMMPS serially
N: Run LAMMPS using N processors. N is internally capped at the total number of configurations in the training set.

## mpiLauncher

*Optional*
*Type:* String
*Range:* None
*Default:* mpiexec
*Description:* MPI launcher used to invoke parallel runs of LAMMPS. This string will be prepended to the LAMMPS invocation command, so it can be used to pass additional options to the launcher (to oversubscribe cores, set processor affinity controls, etc).

## numConstants

*Optional*
*Type:* Integer
*Range:* 1 or 2
*Default:* 2
*Description:* Number of constant terms in the SNAP expansion. By default, a constant term is included for every element. A single constant term can be forced by setting numConstants to 1. If numTypes = 1, this setting is ignored.

## numTypes

*Optional*
*Type:* Integer
*Range:* 1 or 2
*Default:* 1
*Description:* Specify the number of chemical elements in the training set. If set to 1 (or not specified), the following keywords must also be set: rcutfac, rmin0, zblcutinner, zblcutouter, type1, zblz1, radelemen1. If set to 2, the following must be set: rcutfac, rmin0, zblcutinner, zblcutouter, type1, type2, zblz1, zblz2, radelemen1, radeleme2, qcoul, rcoul.

## potentialFileName

*Optional*
*Type:* String
*Range:* None

*Default:* type1+type2
*Description:* Name of the generated SNAP potential files. Fitsnap.py generates three potential files named BASE.snapcoeff, BASE.snapparam, and pot_BASE.mod, where by default BASE is type1+type2 (for numTypes=2), type1 (for numTypes = 1), or whatever the user selects for potentialFileName.

## qcoul

*Required*
*Type:* Floating point
*Range:* $> 0.0$
*Default:* None
*Description:* Atomic charge used in coulombic reference potential. Only required when numTypes = 2. The first species is assigned charge of qcoul, and the second is assigned charge $-$qcoul. Supplying qcoul = 0.0 to deactivate coulombic interactions currently is not supported.

## radelem1 and radelem2

*Required*
*Type:* Floating point
*Range:* $> 0.0$
*Default:* None
*Description:* The contribution of the first element to the SNAP cutoff radius of the first element. The cutoff is determined by summing the element radii and multiplying by rcutfac. If numTypes = 1, then only radelem1 is needed.

## rcoul

*Required*
*Type:* Floating point
*Range:* $> 0.0$
*Default:* None
*Description:* Cutoff distance for the coulombic reference potential. Only required when numTypes = 2.

## rcutfac

*Required*
*Type:* Floating point
*Range:* $> 0.0$
*Default:* None
*Description:* Factor that multiplies the sum of the element radii (see radelem1 and 2) to compute the SNAP cutoff distance.

## rfac0

*Optional*
*Type:* Floating point
*Range:* $> 0.0$
*Default:* 0.99363
*Description:* The constant factor that multiplies $\pi$ to obtain $\theta_0^{max}$. See Equation 2.3 and following.

## rmin0

*Required*
*Type:* Floating point
*Range:* >=0.0
*Default:* 0.0
*Description:* Constant in the linear mapping from radial distance to polar angle. rmin0 is subtracted from both the numerator and denominator in Equation 2.3.

## runLammps

*Optional*
*Type:* Integer (on/off)
*Range:* 0 or 1
*Default:* 1
*Description:* Whether to run LAMMPS. If fitsnap.py was run previously in this location with writeSystem enabled, then the LAMMPS output from that run was serialized in the file DumpSnap/Absystem.dat. With runLammps disabled (0), if fitsnap.py detects this file then fitsnap.py will attempt to use the information it contains in lieu of re-running LAMMPS. (If reading the file fails, LAMMPS will be run, anyway.) If runLammps is enabled, LAMMPS will always be run, regardless of the presence of DumpSnap/Absystem.dat.

## staleOutputCheck

*Optional*
*Type:* Integer (on/off)
*Range:* 0 or 1
*Default:* 0
*Description:* Check whether LAMMPS output is stale. After running LAMMPS, examine the file modification time of all results to ensure that they are recent and not left over from a previous study, which could occur if LAMMPS crashes. (Regardless of this setting, fitsnap.py examines the returncode from the LAMMPS invocation and exits with an error if it is nonzero. staleOutputCheck can be enabled for additional safety.)

## twojmax

*Optional*
*Type:* Integer
*Range:* Any positive, even integer
*Default:* 6
*Description:* Controls the number of bispectrum components in the expansion. See the explanation of J in Section 2.1 for details.

## type1 and type2

*Required*
*Type:* String
*Range:* None
*Default:* None
*Description:* The names of the chemical elements in the training set. If numTypes = 1, then only type1 is needed.

## verifyConfigs

*Optional*
*Type:* Integer (on/off)
*Range:* 0 or 1
*Default:* 1
*Description:* Control whether configurations in LAMMPS output are compared with configurations read in from JSON. Configurations in the training set should match configurations in the LAMMPS output. If not, something unexpected went wrong, and fitsnap.py will exit with an error message. This check is on by default because it ordinarily is inexpensive, but for large configurations or training sets with many configurations, it may be wise to deactivate it.

## wj1 and wj2

*Required*
*Type:* Floating point
*Range:* None
*Default:* None
*Description:* The dimensionless weights in Equation 2.1. Only required when numTypes = 2.

## writeDataKey

*Optional*
*Type:* Integer (on/off)
*Range:* 0 or 1
*Default:* 0
*Description:* Whether to write key mapping training set configurations to LAMMPS data files. Currently, fitsnap.py sorts the configurations by number of atoms in descending order before writing LAMMPS data files. The purpose of the sorting is to achieve greater computational efficiency when LAMMPS is run in parallel. With writeDataKey enabled, a human readable file, Data/datakey.dat, is written, which associates each training set JSON file with its corresponding LAMMPS data file in the Data folder.

## writeSystem

*Optional*
*Type:* Integer (on/off)
*Range:* 0 or 1
*Default:* 0
*Description:* Whether to write several files that may be useful for troubleshooting and reuse by fitsnap.py. They are:

- abtotal.dat: Contains $[A|b|w]$ in plaintext
- DumpSnap/Absystem.dat: Serialization (Python pickle) of LAMMPS output.
- SNAP{energy,force,virial}.dat: The QM prediction, SNAP prediction (without reference potentials contributions), and error for every configuration in the training set for each of the indicated quantities.

- predictions.json: Serialized (JSON) SNAP predictions (including reference potential contributions) of energy, forces, and virials for all training set configurations.

This option is off by default because the files can be large and time-consuming to write. See Section A.7 for further description of their contents.

### zblz1 and zblz2

*Required*
*Type:* Integer
*Range:* $> 0$
*Default:* None
*Description:* The atomic number of each element used by the ZBL reference potential. If numTypes = 1, then only zblz1 is needed.

### zblcutinner and zblcutouter

*Required*
*Type:* Floating point
*Range:* $> 0.0$
*Default:* None
*Description:* Inner and outer radius of the smooth cutoff for the ZBL potential in units of angstroms. That is, the smooth cutoff occurs between zblcutinner and zblcutouter.

## A.5 Required Files

Fitsnap.py expects to find certain files and folders in particular locations relative to where it is run. This section describes these requirements.

### JSON/

The JSON-format training data files must be located in subfolders within a folder named JSON. The subfolders correspond to groups, and the names of the subfolders are the group names, which are also referred to in the group weights file.

### snap/

The snap folder is a Python package and contains code used by fitsnap.py. It can reside either in the same directory as fitsnap.py, or some directory on the user's PYTHONPATH.

### in.snap

This is the static part of the script that directs LAMMPS to calculate the bispectrum components and other information needed by fitsnap.py. (It includes a dynamic portion that fitsnap.py writes every time it runs.) It must be present in folder from which fitsnap.py is run.

### grouplist.in

The list of training set configuration groups and their weights. The format is explained in Section A.6

## A.6 Weights

Fitsnap.py performs a weighted regression as shown in Equation 2.16. Weights are supplied by the user in the file grouplist.in. (The name of the file is optional; see the groupFile keyword.) Unique weights can be applied to each quantity (energy, force, virial) within each user-defined group of training set configurations.

The format of grouplist.in is:

```
# [group name] [# Configs] [Energy Wt.] [Force Wt.] [Virial Wt.]
Group1             25         1000.0       10.0        1.e-8
Group2             50          500.0        5.0        1.e-8
Group3             10        10000.0      100.0        1.e-8
```

The group names must be in alphabetical order, and the names and number of configurations must correspond to the contents of the JSON folder and its subfolders.

## A.7 Expected Output

Fitsnap.py produces the following output.

### SNAP potential files

The three files that define a SNAP potential for use in LAMMPS are written at the end of fitting process. They receive filenames that are either automatically generated by fitsnap.py from the names of the elements or that are specified by the user using the potentialFileName input file keyword. The .mod file is the master, and refers to the other two files, which have the extensions .snapparam and .snapcoeff, by name.

### SNAP{energy,force,virial}_error.dat

Overall and per-group mean fitting errors are reported for each quantity.

### Data/trainingset.dat

Serialized (Python pickle) training set data, which will be used by fitsnap.py on subsequent runs in lieu of re-parsing the JSON-format training set configurations. Note that this file is created every time fitsnap.py is run, and that fitsnap.py does NOT attempt to verify that the training data in JSON/ matches the (de)serialized information. If the training set is changed, it is the user's responsibility to delete the stale Data/trainingset.dat before running fitsnap.py.

### abtotal.dat

Written when writeSystem is enabled. Contains $[A|b|w]$ in plaintext.

### DumpSnap/Absystem.dat

Written when writeSystem is enabled. Serialization (Python pickle) of LAMMPS output. If on

subsequent runs of fitsnap.py, no hyperparameters that affect the calculation of the bispectrum components are changed (i.e. only regression weights), this file can be used to avoid re-running LAMMPS.

### SNAP{energy,force,virial}.dat

Written when writeSystem is enabled. The QM prediction, SNAP prediction (without reference potentials contributions), and error for every configuration in the training set for each of the indicated quantities. The data are in three columns. Within a column, the data are arranged first by configuration in the order shown in Data/datakey.dat. The forces are arranged by atom (in the same order as both the training set and LAMMPS data files) then by x, y, and z component. The virial components are in Voigt order.

### predictions.json

Written when writeSystem is enabled. Serialized (JSON) SNAP predictions (including reference potential contributions) of energy, forces, and virials for all training set configurations.

### Data/datakey.dat

Written when writeSystem is enabled. A key mapping training set configurations to the LAMMPS data and output files in the Data and DumpSnap (See below) directories, respectively.

In addition to these, two folders containing intermediate results are produced.

### Data/

The Data folder contains training set configurations in LAMMPS data file format, which LAMMPS will use to compute bispectrum components and other information. The file names contain 0-padded indices.

### DumpSnap/

LAMMPS writes all log and dump files into this folder as it runs. Upon completion, fitsnap.py reads these files to obtain information it needs to perform the fitting.

## A.8 Caching Tips and Warnings

As stated in the introduction, fitsnap.py caches certain information to avoid needless rework. Caching can be especially beneficial in the context of a Dakota study, which may require evaluation of thousands of distinct sets of hyperparameters.

- Converting the JSON-format training set configurations into LAMMPS data files is a relatively costly operation. If fitsnap.py is run more than once in the same location, it assumes that the training set is unchanged and reuses the files in Data/ that it generated on the previous run. (It bears repeating that it is the user's responsibility to delete stale versions of Data/trainingset.dat when the training set is changed.) If on the other hand separate runs of fitsnap.py that are based on the same training set are performed in different locations (as will be the case when Dakota manages concurrent runs), reconversion can be avoided by

copying or symlinking Data/trainingset.dat folder to a new location. The Data/ folder can be populated initially by running fitsnap.py with the --convert-JSON command line option.

- Similarly, it is unnecessary to re-run LAMMPS when changing only the regression weights, since the weights do not affect generation of the bispectrum components. To reuse LAMMPS output, enable the input file keyword writeSystem and run fitsnap.py once. The file DumpSnap/Absystem.dat, which contains the serialized linear system (without the weights), will be created. To reuse DumpSnap/Absystem.dat, disable writeSystem and runLammps in the input file, and copy or symlink Data/trainingset.dat and DumpSnap/Absystem.dat into your run folder.

# DISTRIBUTION:

| | | | |
|---|---|---|---|
| 1 | MS | 0359 | LDRD Program Office, 07911 |
| 1 | MS | 0754 | R.T. Cygan, 06910 |
| 1 | MS | 0754 | J.A. Greathouse, 06915 |
| 1 | MS | 0754 | S. Teich-McGoldrick, 06915 |
| 1 | MS | 0899 | J.M. Lane, 01814 |
| 1 | MS | 1056 | W.R. Wampler, 01111 |
| 1 | MS | 1159 | R.J. Bondi, 01344 |
| 1 | MS | 1179 | L.J. Lorence, 01341 |
| 1 | MS | 1189 | M.P. Desjarlais, 01600 |
| 1 | MS | 1189 | T.R. Mattsson, 01641 |
| 1 | MS | 1303 | G.S. Grest, 01131 |
| 1 | MS | 1315 | N.A. Modine, 01131 |
| 1 | MS | 1315 | J.S. Nelson, 01131 |
| 1 | MS | 1315 | M.J. Stevens, 01814 |
| 1 | MS | 1322 | B.A. Hendrickson, 01420 |
| 1 | MS | 1322 | J.B. Aidun, 01425 |
| 1 | MS | 1322 | J.E. Moussa, 01425 |
| 1 | MS | 1322 | R.J. Hoekstra, 01426 |
| 1 | MS | 1322 | C.R. Trott, 01426 |
| 1 | MS | 1322 | S.S. Collis, 01440 |
| 1 | MS | 1322 | J.A. Stephens, 01441 |
| 1 | MS | 1322 | J.R. Stewart, 01441 |
| 1 | MS | 1322 | L.P. Swiler, 01441 |
| 1 | MS | 1322 | S.D. Bond, 01442 |
| 1 | MS | 1322 | P.S. Crozier, 01444 |
| 1 | MS | 1322 | R.J. Magyar, 01444 |
| 1 | MS | 1322 | S.G. Moore, 01444 |
| 1 | MS | 1322 | S.J. Plimpton, 01444 |
| 1 | MS | 1322 | P.A. Schultz, 01444 |
| 1 | MS | 1322 | A.P. Thompson, 01444 |
| 1 | MS | 1322 | V. Tikare, 01444 |
| 1 | MS | 1322 | A.E. Wills, 01444 |
| 1 | MS | 1411 | F.F. Abdeljawad, 01814 |
| 1 | MS | 1411 | S.M. Foiles, 01814 |
| 1 | MS | 1411 | T-R. Shan, 01814 |
| 1 | MS | 1411 | A.C. Sun, 01814 |
| 1 | MS | 1415 | K. Leung, 01131 |
| 1 | MS | 1415 | A.F. Wright, 01131 |
| 1 | MS | 9957 | R.E. Jones, 08256 |

| | | | |
|---|---|---|---|
| 1 | MS | 9957 | X. Zhou, 08256 |
| 1 | MS | 9957 | J.A. Zimmerman, 08256 |
| 1 | MS | 9957 | J.A. Templeton, 08365 |
| 1 | MS | 9957 | G.J. Wagner, 08365 |
| 1 | MS | 0899 | Technical Library, 9536 (electronic copy) |

Sandia National Laboratories