Improving Material Simulation Efficiency by Improving Interpolation Schemes

Spencer E. Hart

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Gus L. W. Hart, Advisor

Department of Physics and Astronomy

Brigham Young University

April 2016

ABSTRACT

Improving Material Simulation Efficiency by Improving Interpolation Schemes

Spencer E. Hart
Department of Physics and Astronomy, BYU
Bachelor of Science

High-throughput alloy simulations can greatly increase the rate at which we discover and synthesize new materials by giving narrower focus and clearer direction to physical materials experimentation. In working towards a comprehensive database of potential alloys and their predicted characteristics, we are seeking ways to increase the computational efficiency of our simulations. One main opportunity for improvement is in calculating the energy contribution from electron bands. Determining this energy contribution requires numerically integrating over the occupied regions of the electron bands. For metals in particular, dense sampling of the electron bands is required to achieve sufficient accuracy in the integral (due to the lack of smoothness in the partially filled electron bands of metals). Each sample point requires solving a large eigenvalue problem, leading to longer computation time for denser sampling. This thesis describes attempts to interpolate the electron bands using trigonometric star functions and splines to achieve necessary accuracy with sparser sampling. The findings I present here show that the interpolation methods we have employed do not represent the bands well enough to be used to reduce sampling of the electron bands.

Keywords: DFT, Integration, Electron, Bands, Interpolation, Splines, High-Throughput, Simulations

ACKNOWLEDGMENTS

I would like to give my thanks to the following people: my amazing older brother and advisor, Dr. Gus L. W. Hart, for being someone I can look up to in all aspects of life, including physics; Jeremy Jorgensen and Matt Burbidge for putting up with my strong personality while we worked on this project; Derek Thomas for laying the foundation of the spline aspect of this project; and my wife for her patience, support, and editing during the writing process.

I would also like to thank the Office of Naval Research and the BYU Department of Physics and Astronomy for providing funding.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Overview

The development and use of new materials is vital to the continual progress of science and technology. New materials have been crucial in shaping past civilizations, hence the names “Bronze Age” and “Iron Age”. In the 20^{th} century, the development of suitable semiconductor materials allowed for the dawn of the “Silicon Age”. As we seek to further the reaches of science and technology in the 21^{st} century, bright minds are constantly coming up with novel ways to use the materials we already have. Often, however, new ideas and inventions are inhibited by a lack of ideal materials for their implementation. As we stretch the boundaries of what is possible with the materials we already have, it becomes increasingly important to discover and develop new materials. The process of discovering and developing, unfortunately, tends to be very slow (in some instances taking more than 20 years before the material was commercially available). In light of this, the White House has issued the Materials Genome Initiative (MGI), which pushes to increase the rate at which new materials are discovered, developed, and deployed [1].

In accordance with the MGI, our research group is concerned with the discovery and invention of new metal alloys through high-throughput (HT) computational methods. This is primarily accomplished with a program, AFLOW [2, 3], automatically running and analyzing results from Density Functional Theory (DFT) simulations. Currently, the AFLOW database contains "1,140,836 material compounds - with over 102,675,240 calculated properties (and growing)" [4]. The database represents more than 100 million CPU hours using BYU's Fulton Supercomputing Lab and additional CPU time on other supercomputers. Based on the \$0.12/core/hour Sabalcore Computing, Inc. charges for supercomputer rental, this amounts to over \$12 million worth of supercomputer time for these simulations. Any increase in computational efficiency would reduce the CPU time needed for each simulation, saving time and resources and allowing us to increase our HT productivity.

This thesis deals specifically with applying interpolation methods to DFT calculations in an attempt to increase convergence rates of electron band energy calculations. It will be organized as follows: Chapter 1 gives background into alloy simulations using DFT, specifically focusing on the electron bands and the difficulties they present. Chapter 2 presents the different interpolation techniques we applied to electron band data. Chapter 3 presents the results of the interpolations, the issues that were encountered, and attempts to resolve them. Finally, our conclusions on the interpolation methods and plans for future work will be presented in Chapter 4.

1.2 Alloy Simulations

1.2.1 Crystalline Structure

Simulating alloys first requires an understanding of the general structure of metals. Metals are crystalline in structure, meaning they are very ordered and periodic. Crystalline structures are described in terms of Bravais lattices (often just called lattices). Simply put, a lattice is a repeating grid of points. Figure 1.1 shows examples of each of the five fundamental 2D Bravais lattices.

Each lattice point (the green circles) represents an equivalent point in space (equivalent atom in our case). The vectors between lattice points (denoted by $\mathbf{a_1}$ and $\mathbf{a_2}$) are called lattice vectors. Since lattice points are equivalent, any integer combination of lattice vectors define a valid period $\mathbf{T}$ of the lattice. (This concept is very important in section 2.1.2.) Each type of lattice also has specific rotational symmetries associated with it: 90°, 180°, and 270° for a square lattice, but only 180° for the rectangular lattice. While 2D lattices are helpful for illustration, we are interested in 3D materials. In 3D, there are 14 unique types of Bravais lattices (three examples are shown in Figure 1.2). The same principles and terminology that apply to 2D lattices also apply to 3D lattices, though with an additional lattice vector ($\mathbf{a_3}$) and up to 48 rotational symmetry operations. *Solid State Physics* by Harold Stokes [5] contains definitions and details for each type of 3D lattice.

In a pure metal, each lattice point would represent the same element. In alloys, however, impurities are intentionally added to confer specific characteristics [7], such as the high impact strength achieved in mangalloy by adding approximately 13% manganese to steel with about 1% carbon. The addition of impurities can affect the crystalline structure in several ways. In *solid solutions*, impurity atoms are randomly and uniformly distributed throughout the more abundant element (called the solvent). The alloy simulations we run, however, are usually intermetallic compounds rather than solid solutions. Intermetallic compounds are characterized by "distinct chemical formulas" [8], meaning there is a particular ratio of two or more metals. The different metals together form a distinct unit cell that is repeated throughout the material. I will limit further discussions to this type of alloy.

1.2.2 Thermodynamic Stability

To determine whether or not a particular alloy will form, we have to ascertain if it is thermodynamically stable. This requires calculating the ground-state energy of pure metals and the alloys being simulated by solving Schrödinger's equation—a very daunting task when you consider that

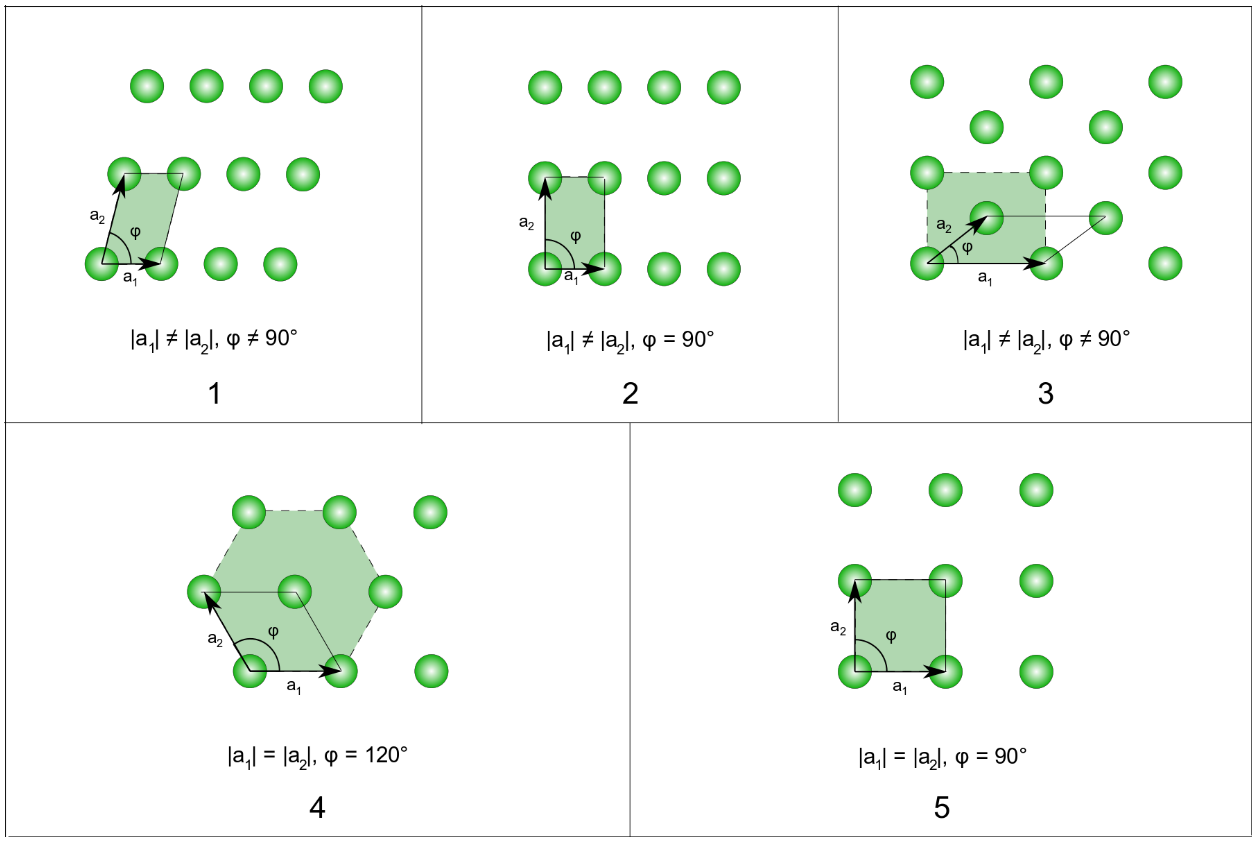

Figure 1.1 The five fundamental 2D Bravais lattices: 1 oblique, 2 rectangular, 3 centered rectangular, 4 hexagonal, and 5 square. [6]

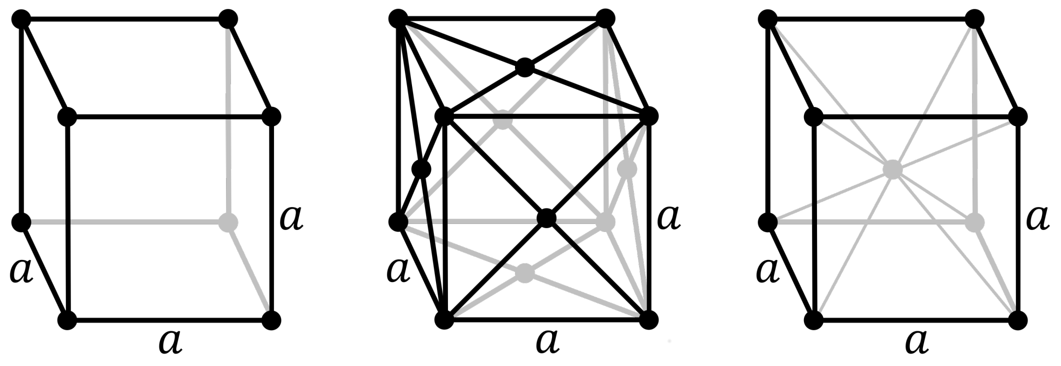

Figure 1.2 The three cubic Bravais lattices. From left to right: simple-cubic, face-centered cubic, body-centered cubic. [6]

Schrödinger's equation for a nanocluster of 100 Aluminum, with 1300 electrons, is a differential equation with over 4000 variables! Density Functional Theory (covered in the following subsection) is an effective way to find an approximate solution of Schrödinger's equation for many-body systems like alloys. The following example illustrates how the alloy simulations work once you have a method to calculate the ground-state energy.

If the alloy being simulated is a binary alloy (containing only two elements, which we will arbitrarily call A and B), the ground-state energies of pure element A and pure element B are first computed independently. The ground-state energy is next computed at a various A/B ratios for each type of lattice. The energies of all the various A/B mixtures are compared against each other and the energies of the pure elements. If none of the A/B mixtures have a ground-state energy lower than the pure elements' ground-state energies, the elements will separate into regions of pure A and regions of pure B (since formation of the pure metals is more favorable than the mixture). Otherwise, the A/B mixtures with the lowest energies are thermodynamically stable and likely to form.

The same general technique is applied to alloys with three or more constituents. There is the added complication, however. Rather than only comparing the ternary A/B/C mixtures to each other and the pure elements, the ternary alloys must also be compared with the possible binary mixtures (A/B, B/C, and A/C). Quaternary alloys must be compared with all possible ternary and binary alloys as well as the pure elements. The pattern would continue for alloys with more constituents.

1.2.3 Density Functional Theory (DFT)

One method to "approximate solution[s] to the many-body Schrödinger equation" is DFT [9]. A detailed explanation of DFT is outside the scope of this thesis. However, a brief description is in order, since we are attempting to increase efficiency in certain aspects of DFT calculations, particularly when simulating metals and alloys. For a more detailed explanation of DFT, I defer the reader to works dedicated to the topic, such as *Density Functional Theory: A Practical Introduction* by Sholl and Stechel [10].

DFT is based on two theorems: 1) "The ground-state energy from Schrödinger's equation is a unique functional of the electron density;" 2) "The electron density that minimizes the energy of the overall functional is the true electron density corresponding to the full solution of the Schrödinger equation" [10]. The first theorem indicates that a particular electron density and a functional relationship is all that is needed to specify a ground-state energy. While the exact functional relationship between the electron density and ground-state energy is unknown, there are ways to approximate it implemented in DFT methods. The second theorem indicates that once we know the functional relationship between the electron density and the energy, we need to vary the electron density until we find the one that minimizes the energy.

While this general approach used in DFT can be stated succinctly, the details quickly become complex. There are many available software packages that implement DFT methods (and take care of most of the complex details automatically). We primarily use the Vienna Ab initio Simulation Package (VASP) [9]. It is one of the most well-known codes for running DFT simulations.

1.2.4 DFT Accuracy and Efficiency Factors

Since determining alloy stability requires finding the difference between energies, which can be quite small, we need our calculations of the ground-state energies to be rather accurate. In general, our desired accuracy is within 10^{-3} eV per atom. There are several factors that contribute to the total ground-state energy, including "electron kinetic energies, the Coulomb interactions between the electrons and the nuclei, the Coulomb interactions between pairs of electrons, and the Coulomb interactions between pairs of nuclei" [10].

Most of the contributions to the ground-state energy can be quickly determined once the elec-

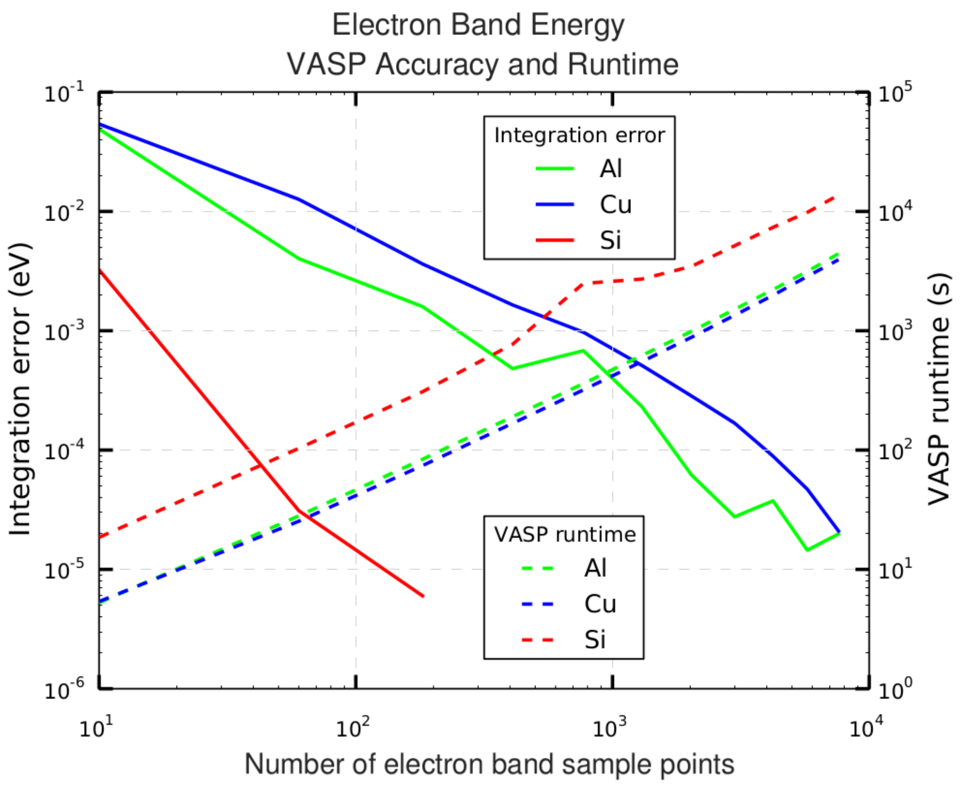

Figure 1.3 The electron band energy for metals converge slowly, requiring dense electron band sampling and long run-times to achieve errors below 10^{-3} eV per atom. (Compare this to semiconductors, like silicon, that require many fewer sample points for equivalent accuracy.) This figure is a courtesy of Jeremy Jorgensen.

tron density has been solved for. However, the energy contribution of each electron in the potential of the atomic nuclei and all other electrons is computationally expensive to determine accurately; the energies of all the occupied electron states must be added together. In alloys, the allowed electron energy states become a complicated, continuous, multivalued function, called the *electron bands* (discussed in detail in the following section). To determine the energy contribution of the electron bands, DFT codes integrate over the occupied states. We do not have access to an analytic form of the electron bands, so numerical integration techniques must be used. As with any numerical integration, the accuracy of the result depends largely on the coarseness of the function sampling; when greater accuracy is needed, simply increase the number of sample points. However, getting electron band values through DFT is computationally expensive. Figure 1.3 shows how the run-time increases with the number of electron band sample points.

Particularly in metals, the accuracy of electron band energy calculations converge slowly with the simple numerical integration. Since this calculation accounts for the greatest variation in simulation run-times, we are focusing our efforts here to increase computational efficiency. Chapter 2 explores two different interpolation methods we investigated for their potential to achieve greater accuracy with fewer sample points.

1.3 Electron Bands

In the Bohr model of atoms, we learn that electrons can only occupy certain discrete energy levels. As atoms come together to form bonds, such as in molecular hydrogen (H_2), the energy levels in each atom change slightly—one up, and one down. This effect is called orbital splitting.

Each of these new molecular orbitals can be described by combining atomic orbitals and identified by a wavelength of the new orbital function envelope. If all the orbitals are combined constructively, the effective wavelength is equal to the length of the molecule or group of atoms. In bulk materials, such as metals, there are enough atoms that this wavelength is essentially infinite in comparison to the atomic scale. If all the orbitals are destructively interfering, the wavelength is equal to the width of an individual atom, called the *lattice constant* or a in crystalline materials. Rather than identifying each energy level by its wavelength, however, it is more convenient to use the wave number

$$k = \frac{2\pi}{\lambda}. \tag{1.1}$$

Since the effective wavelength (λ) of each energy level lies between infinity and a, expressed in wave numbers, the domain of all possible energy states in the material is

$$0 \leq k \leq \frac{2\pi}{a}. \tag{1.2}$$

This domain is referred to as *k-space* or *reciprocal space*, since k is proportional to the reciprocal of the real-space variable λ. In three dimensions, k will become a vector $\mathbf{k}$, and the domain becomes a volume (called the Brillouin zone) defined by three vectors, $\mathbf{b_1}$, $\mathbf{b_2}$, and $\mathbf{b_3}$, called *reciprocal lattice vectors*. These reciprocal lattice vectors are specific to the particular lattice of the alloy or metal being described. For example, the reciprocal lattice vectors of a face-centered cubic lattice always define a body-centered cubic lattice and vice versa. Reciprocal lattice vectors will come up again in a different context in section 2.1.2.

If there are only a few atoms being considered, the energy function is still only defined at discrete points in the domain of $\mathbf{k}$. However the combination of many, many atoms in bulk materials leads to enough orbital splitting that the formerly discrete atomic energy levels become a collection of continuous functions, denoted electron bands. Similar to the multiple allowed energy states in a single atom (1s, 2s, 2p, etc. orbitals), every point in the domain of $\mathbf{k}$ (the Brillouin zone) also has multiple allowed energy states (hence the plural electron *bands*). DFT codes calculate the different allowed energy levels at explicitly specified points or sample grids such as Monkhorst-Pack [11]. The energy values for each point are returned, sorted by size: the lowest energy value from all points is designated the first band, the collection of next lowest points the second band, etc.

Electron band functions are notoriously hard to plot and visualize not only because they are multivalued, but because of their 3D domain. Often, *spaghetti plots* are used, in which the band values are determined along various straight-line paths between points with high symmetry. Figure 1.4 shows an example of one of these plots for pure aluminum on the right and the corresponding high-symmetry points pictured in an FCC Brillouin zone on the left.

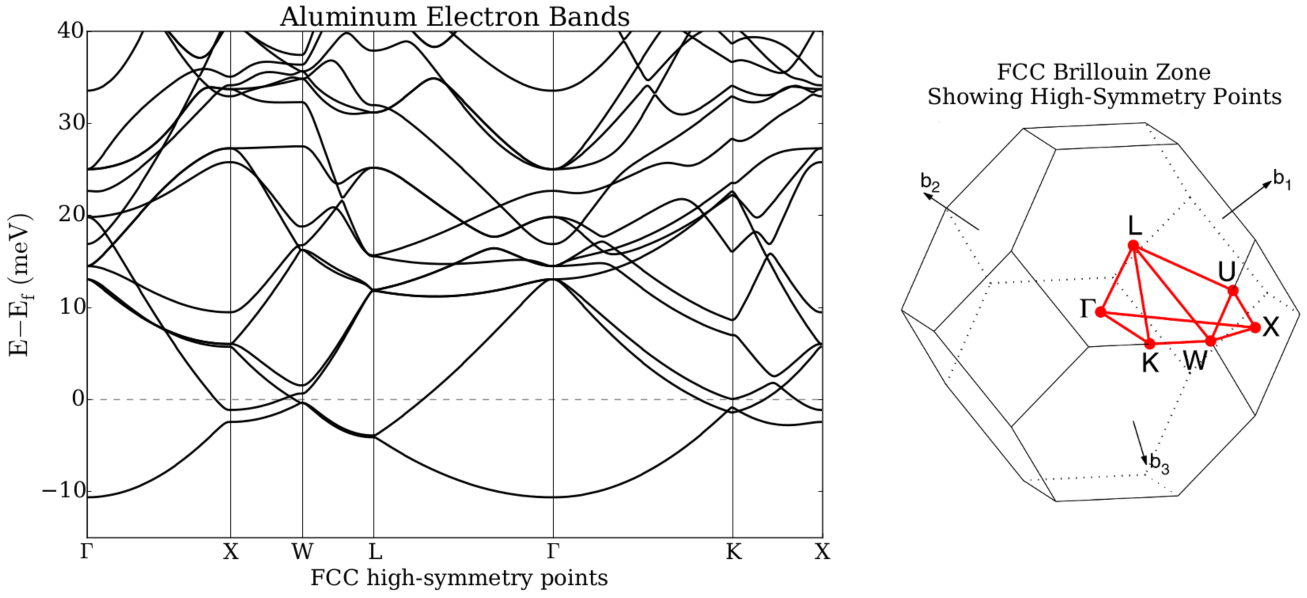

Figure 1.4 *Left:* High-symmetry paths of aluminum electron bands where each panel is a different path. This is sometimes called a *spaghetti plot*. *Right:* Corresponding Brillouin zone with the high-symmetry points labeled [12].

1.3.1 Implications of Multivalued Bands

The multivalued nature of the electron bands presents a concern when trying to create an interpolation. Since our interpolation schemes only fit single-valued functions, we must treat the electron bands as multiple single-valued functions and interpolate each one individually. The simplest approach uses the convention mentioned earlier: all of the lowest energy values are the first function, all of the second-lowest energy values are the second function, etc. (Results using this method are presented in sections 3.1 and 3.2.) While separating the bands this way is easy, the functions that result are not very smooth. To illustrate, I have added additional spacing between the energy values at each sample point in Figure 1.5. The lack of smoothness requires more sample points to build an accurate interpolation.

In theory, smooth functions could be used to represent each individual band—there are visually apparent smooth paths between each pair of symmetry points in Figure 1.4. Smooth functions are

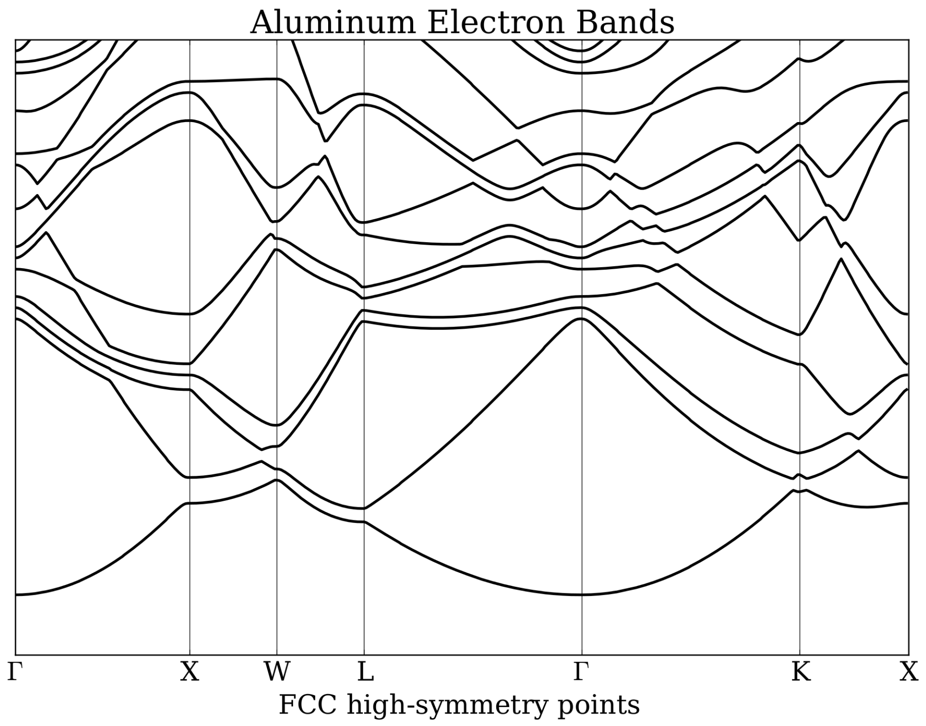

Figure 1.5 High-symmetry paths of aluminum electron bands are shown with additional spacing added between the energy values at each point. This demonstrates the lack of smoothness when electron bands are separated into individual functions by simply ordering the energy values at each point from least to greatest.

easily interpolated without many sample points, which is good for HT efficiency. It is difficult, however, to arrange the sampled energy values so that you can individually interpolate the visually apparent smooth bands. This is especially true when dealing with sparse sample grids (the sampling used to create the Figure 1.4 was very dense). Section 3.3 discusses the methods we employed to untangle the individual smooth bands from one another.

1.3.2 Periodicity of Bands

One key feature of electron band functions is that they have a repeating unit cell, as do the metals that generate them. Because of this, the discussion about crystalline structure in section 1.2.1

also applies to the electron bands. While the electron bands are defined on a reciprocal lattice, which is generally different from the lattice of the metal, the electron bands exhibit all of the symmetry of the metal that generated them. VASP exploits this fact by reducing the Brillouin zone sample points using the symmetry operations of the metal. The periodic nature of the electron band functions motivated the selection of both interpolation methods described in the following chapter.

Chapter 2

Interpolation Methods

2.1 Trigonometric Star-Function Basis

2.1.1 1D Fourier Expansion

Undergraduates in physics learn early on in their coursework about using Fourier expansions to approximate 1D periodic functions as a sum of sine and cosine functions or, equivalently, complex exponential functions:

$$f(x) \approx a_0 + \sum_{n=1}^{N} a_n \cos(\frac{2\pi nx}{L}) + b_n \sin(\frac{2\pi nx}{L}), \tag{2.1}$$

$$f(x) \approx \sum_{n=-N}^{N} c_n \exp(\frac{i2\pi nx}{L}), \tag{2.2}$$

where N specifies the number of terms in the expansion. (Notice that since each basis function in the sums above is periodic in x with period L, the approximation of f must be as well—i.e. $f(x+L) = f(x)$.) An approximation of f can be defined by choosing N and determining appropriate values of the coefficients a_n and b_n or c_n. If an analytical form of f is known, the coefficients can be calculated exactly using Fourier's trick (i.e. exploiting orthonormality) [13].

However, if an analytic form is unavailable—either because it is not known or does not exist—the coefficients can be determined by sampling the function at various points and solving an over-determined matrix equation of the form $\mathbf{Ax} = \mathbf{b}$. If the Fourier expansion is expressed in complex exponential form, the matrix equation is as follows:

$$\begin{bmatrix} \exp(\frac{i2\pi(-n)x_1}{L}) & \exp(\frac{i2\pi(1-n)x_1}{L}) & \cdots & \exp(\frac{i2\pi(n-1)x_1}{L}) & \exp(\frac{i2\pi(n)x_1}{L}) \\ \exp(\frac{i2\pi(-n)x_2}{L}) & \exp(\frac{i2\pi(1-n)x_2}{L}) & \cdots & \exp(\frac{i2\pi(n-1)x_2}{L}) & \exp(\frac{i2\pi(n)x_2}{L}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \exp(\frac{i2\pi(-n)x_m}{L}) & \exp(\frac{i2\pi(1-n)x_m}{L}) & \cdots & \exp(\frac{i2\pi(n-1)x_m}{L}) & \exp(\frac{i2\pi(n)x_m}{L}) \end{bmatrix} \begin{bmatrix} c_{-n} \\ c_{1-n} \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{2.3}$$

where $m > n$. Each row in the first matrix contains all the basis functions used in the expansion evaluated at a particular sample point. The first column vector, $(c_{-n}, c_{1-n}, \dots, c_{n-1}, c_n)$, contains the unknown coefficients we are trying to determine. The column vector on the right-hand side contains the actual function values at the sample points. We can then use a computer program, such as NumPy [14], to solve for the coefficients with the least squares method.

2.1.2 3D Fourier Expansion

The same principles and techniques used in 1D Fourier expansions carry over into higher dimensional spaces, but with the added complication of considering in what *direction* ($\mathbf{T}$) the function is periodic:

$$f(\mathbf{r}+\mathbf{T}) = f(\mathbf{r}). \tag{2.4}$$

As mentioned in section 1.2.1, for a function defined on a particular lattice (with lattice vectors $\mathbf{a_1}$, $\mathbf{a_2}$, and $\mathbf{a_3}$) any period $\mathbf{T}$ can be expressed as

$$\mathbf{T} = (m_1\mathbf{a_1}, m_2\mathbf{a_2}, m_3\mathbf{a_3}) \tag{2.5}$$

where $m_1, m2, m3 \in \mathbb{Z}$. In order to approximate such a function with Fourier expansion, the basis functions need to be periodic in $\mathbf{T}$ as well. In 1D, the factor $i2\pi n/L$ ensured the basis functions were all periodic in L. For higher dimensions, we will replace this factor with an as yet undefined vector $\mathbf{G}$:

$$g(\mathbf{r}) = \exp(i\mathbf{G}\cdot\mathbf{r}), \tag{2.6}$$

$$g(\mathbf{r}+\mathbf{T}) = \exp(i\mathbf{G}\cdot(\mathbf{r}+\mathbf{T})) = \exp(i(\mathbf{G}\cdot\mathbf{r}+\mathbf{G}\cdot\mathbf{T})) = \exp(i\mathbf{G}\cdot\mathbf{r})\exp(i\mathbf{G}\cdot\mathbf{T}). \tag{2.7}$$

Now, as long as we choose $\mathbf{G}$ such that $\mathbf{G}\cdot\mathbf{T} = 2\pi n$, where $n \in \mathbb{Z}$, the last exponential in equation 2.7 reduces to one, and we have shown that

$$g(\mathbf{r}+\mathbf{T}) = g(\mathbf{r}). \tag{2.8}$$

To determine appropriate $\mathbf{G}$ vectors for a lattice with lattice vectors $\{\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}\}$, *reciprocal lattice vectors* $\{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}\}$ are defined such that

$$\mathbf{a_i}\cdot\mathbf{b_j} = 2\pi\delta_{ij}. \tag{2.9}$$

While $\mathbf{b_j}$ vectors can be expressly determined for any set of $\mathbf{a_i}$ vectors, this is unnecessary for determining appropriate $\mathbf{G}$ vectors as long as we express $\mathbf{r}$, and hence $\mathbf{T}$, in terms of lattice vectors. Choosing

$$\mathbf{G} = (n_1\mathbf{b_1}, n_2\mathbf{b_2}, n_3\mathbf{b_3}), \tag{2.10}$$

where $n_1, n_2, n_3 \in \mathbb{Z}$, and using equations 2.5 and 2.9 yields

$$\mathbf{G}\cdot\mathbf{T} = 2\pi(n_1m_1 + n_2m_2 + n_3m_3) = 2\pi n', \tag{2.11}$$

where $n' \in \mathbb{Z}$. Equation 2.11 shows that our condition for periodicity in $\mathbf{T}$ in equation 2.7 is met by our choice of $\mathbf{G}$ in equation 2.10. By iterating through all permutations of three integers for n_1, n_2, and n_3, we develop a set of $\mathbf{G}$ vectors that account for the directions and frequencies in which our function is periodic. We can now define our 3D Fourier expansion:

$$f(\mathbf{r}) \approx \sum_{\mathbf{G}} c_{\mathbf{G}} \exp(i\mathbf{G}\cdot\mathbf{r}) = \sum_{-N_1}^{N_1}\sum_{-N_2}^{N_2}\sum_{-N_3}^{N_3} c_{\mathbf{G}} \exp(i2\pi(n_1\alpha_1 + n_2\alpha_2 + n_3\alpha_3)) \tag{2.12}$$

where **r** in expressed in lattice coordinates: $(\alpha_1\mathbf{a_1}, \alpha_2\mathbf{a_2}, \alpha_3\mathbf{a_3})$, $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}$.

Electron band sampling in VASP is usually done in lattice coordinates anyway, so there is no need to transform **r** after extracting it from VASP outputs. At this point, we can use equation 2.12 and sample points to populate the left-hand side matrix in equation 2.3 (each column corresponding to a particular $\{n_1, n_2, n_3\}$, and each row corresponding to a particular sample point). First, however, we want to form star functions to enforce the rotational symmetry of the electron bands.

2.1.3 Star Functions

By approximating the electron bands using equation 2.12, we can ensure that the periodic nature of the bands is maintained. However, the 3D Fourier expansion does nothing to enforce the rotational symmetry of the bands. To enforce rotational symmetry, we collect all of the basis functions that are symmetrically equivalent into one term with one coefficient. The resulting function is called a *star function*, since each term contains multiple components that each point in a different direction.

As part of the DFT simulations, VASP outputs rotation matrices for the material being simulated. The matrices represent all applicable rotation symmetries. By applying each matrix operation to a particular (n_1, n_2, n_3) triplet (representing a unique **G** vector), we generate all equivalent triplets and group them. The process is repeated with a new, unvisited (n_1, n_2, n_3) triplet until the desired number of unique triplet groups—each defining a star function—is achieved. Once the star-functions are all determined, they can be used to populate a matrix (like in equation 2.3) where each column contains a unique star function and the rows correspond to sample points. Least squares methods can then be employed to solve for appropriate coefficients.

2.1.4 dftintegrate (Python module)

I assisted Matt Burbidge in assembling a Python 3 module to implement trigonometric star functions as an interpolation scheme for electron band data from VASP. It includes data extraction from the VASP output files, fitting electron band data to a star function expansion, and evaluation and integration of the fit. The module is available publicly in Python through *pip* (using the command "pip install dftintegrate").

Currently, dftintegrate does not employ analytic integration methods to integrate only a portion of the domain. It instead integrates using a 3D equivalent of the rectangle method. The integral for the entire domain, however, is easily obtained from the fit, since it is simply the c_0 coefficient (all other terms integrate to 0 over a full period).

The performance of this module on two toy problems and actual DFT data are reported in Chapter 3.

2.2 Periodic Spline Basis

A spline interpolation method uses piece-wise polynomials to construct basis functions. The polynomials are joined together at locations called knots in such a way that the overall basis function retains a high level of smoothness between the individual polynomials. For example, the first and second derivatives of adjacent polynomials can be equated at the knot that connects them. The left of Figure 2.1 shows a 1D example where four 3^{rd}-degree polynomials have been joined together. There are many different ways to choose the degree of polynomials, where they are joined, etc. Since the electron band functions that we are trying to interpolate are periodic, I will specifically address spline basis functions that are also periodic.

In periodic spline basis functions, the knots, or adjoining points, are uniformly spaced. The individual basis functions are all constructed using the same piece-wise polynomials, but shifted to other knot intervals (illustrated by the right of graph in Figure 2.1). The number of basis functions is equal to the number of unique knots specified; because we are dealing with periodic functions, knots 1 and 6 in Figure 2.1 right are equivalent, implying we need five basis functions. Each basis

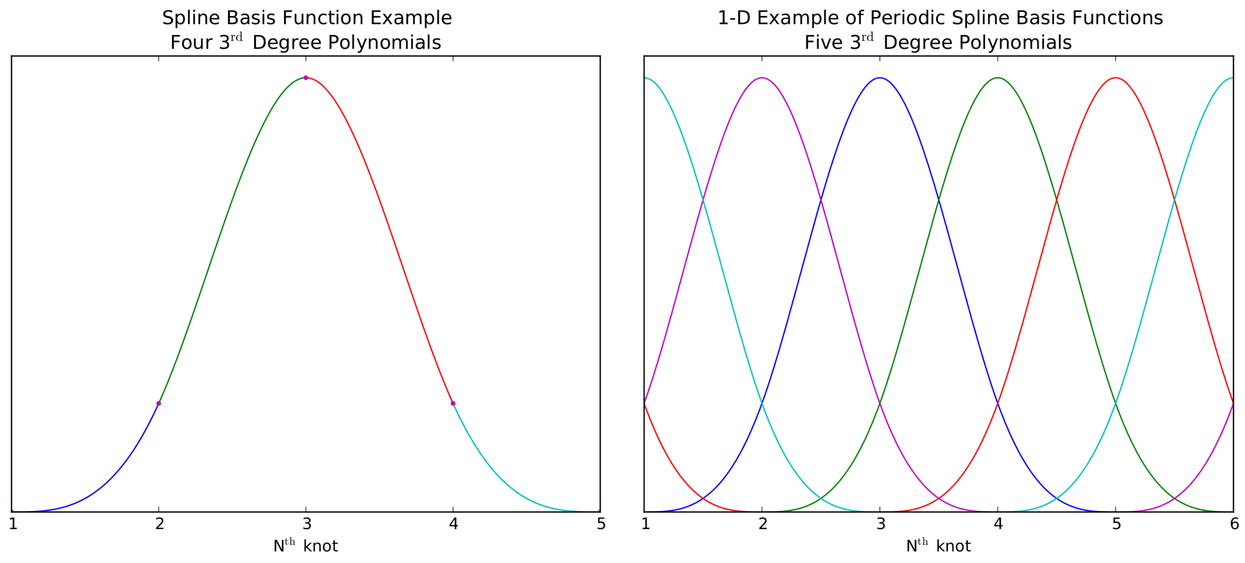

Figure 2.1 *Left:* Example of a single periodic spline basis function. Each colored segment represents a distinct 3rd-degree polynomial. The marked points are the knots. *Right:* Example of a set of 5 basis functions. Each is a shifted form of the piece-wise polynomial on the left.

function has a $d+1$ non-zero segments, or elements, where d is the degree of the polynomials used.

In higher dimensions, there is a *knot vector* for each dimension. For example, the first knot vector, $(x_1, x_2, x_3, \ldots, x_N)$, would be equally spaced knots in the x dimension (with x_1 and x_N being equivalent, since we are dealing with periodic basis functions). Similar vectors would be chosen for the y and z dimensions. In one dimension, each basis function consisted of individual polynomial segments for each unique knot. In higher dimensions, each unique combination of x, y, and z coordinates corresponds to a distinct, multi-dimensional, polynomial, denoted an *element*. Further details about generalizing to higher dimensions are beyond the scope of this paper.

To determine the appropriate coefficients for the interpolation, each element of the spline needs a sample data-point. The sample points either coincide with the knots or centered between them (depending on the parity of the degree of the polynomials being used). Therefore, particular set of knot vectors, combined with the degree parity, determines the sampling scheme for that interpolation. In our case, we decide on the uniform sampling scheme and polynomial degree first, and then we use these to determine the knot vector.

A Monkhorst-Pack grid [11] is generally used for sampling the Brillouin Zone in VASP and many other DFT codes. Monkhorst-Pack grids are uniform in k-space, which is exactly what is needed for periodic spline interpolations. However, VASP also uses the symmetry of the alloy to reduce the uniform grid to a set of symmetrically distinct points. While this improves calculation time by reducing the number of points evaluated, it also introduces difficulties in using a spline interpolation, since the full uniform grid is needed for the spline interpolation. To recover a uniform grid, we can reapply the symmetry operations to the reduced grid and include the new points this produces. However, in non-SC structures this also introduces extraneous points not part of the original Mankhorst-Pack grid. These extraneous points must removed from the sample before proceeding. In order to avoid these complications during our spline interpolation testing phase, we did not use the built in Monkhorst-Pack grid, as there is no option in VASP to disable the symmetry reduction. Instead, we generated a Monkhorst-Pack sampling grid independently and used VASP to evaluate the bands at each point explicitly.

2.2.1 ksi (Julia package)

Previous work had been done by Derek Thomas to produce a Python module for building spline interpolations. The original version, used as a proof-of-concept, only accounted for up to two dimensions and suffered from performance issues. Jeremy Jorgensen and I rebuilt the original Python module in a Julia package called **ksi** and extended the code to account for up to three dimensions. The Julia package is stored in a private GitHub repository owned by Dr. Gus Hart.

Chapter 3

Interpolation Results

3.1 Trigonometric Interpolation with dftintegrate

3.1.1 Single-valued, Smooth, Periodic Toy Function

The simplest case I tested the dftintegrate module on was the function:

$$W1(\mathbf{r}) = \exp(\cos(2\pi x) + \cos(2\pi y) + \cos(2\pi z)). \tag{3.1}$$

This function is convenient to work with since it is defined on a simple cubic lattice where $\{\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}\} = \{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$; the lattice coordinates are simply Cartesian coordinates. I generated two types of sample data sets from this function for testing. The first (referred to as *unreduced*) is an even sampling of the unit cell in all dimensions. For the second type (referred to as *reduced*), any symmetrically equivalent duplicates were removed. This latter set more accurately reflects VASP DFT calculations, where the primary mode of sampling also involves a symmetry reduced set of points. I tested each type of sampling scheme with varying degrees of coarseness, from a maximum of 3 divisions in each dimension to a maximum of more than 40. After formatting the data to be read in by dftintegrate, I created a trigonometric star-function fit of equation 3.1 using each sampling

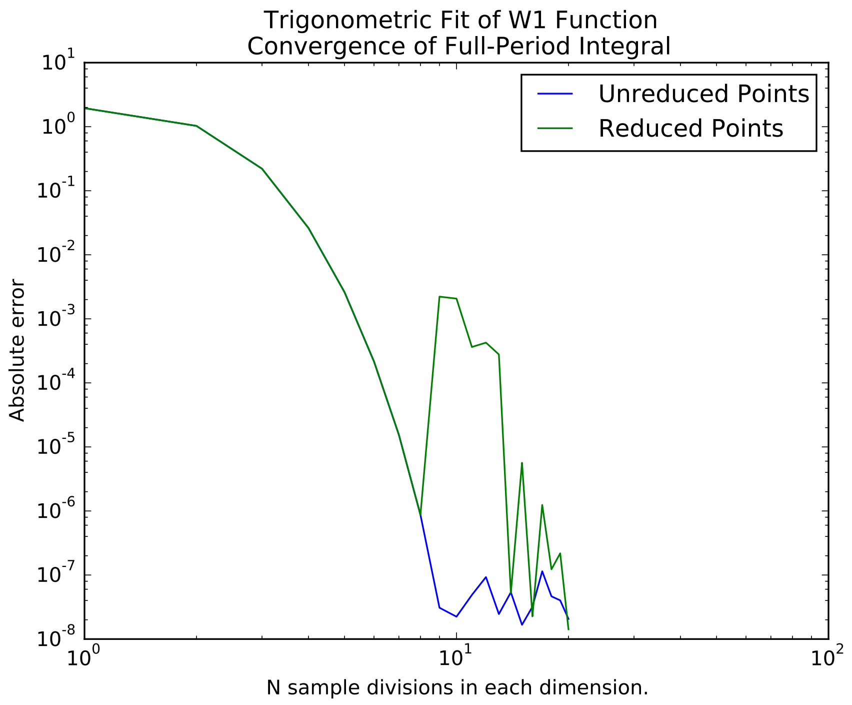

Figure 3.1 Full-period integral convergence of trigonometric fits of $W1$ generated with increasing sample points.

scheme and density.

The first method I used to quantify the goodness of the fit was integrating over the entire unit cell. The value of this integral for the fit is trivial to obtain; it is simply the c_0 coefficient value (since all other terms integrate to zero when integrating a full period). By integrating equation 3.1 in Mathematica exactly, I was able to calculate an absolute error. Figure 3.1 shows the absolute error as a function of sample divisions in each dimension.

While the integral does converge to well below the 10^{-3} error limit we are looking for, this plot shows two unexpected outcomes: 1) since the basis functions enforce all of the symmetry of the toy function, we do not expect differences between fits generated with the reduced and unreduced sample points; 2) we expect the absolute error to converge until double precision at 10^{-16}, but instead it plateaus at 10^{-8}. We have not yet determined the reason for either of these outcomes.

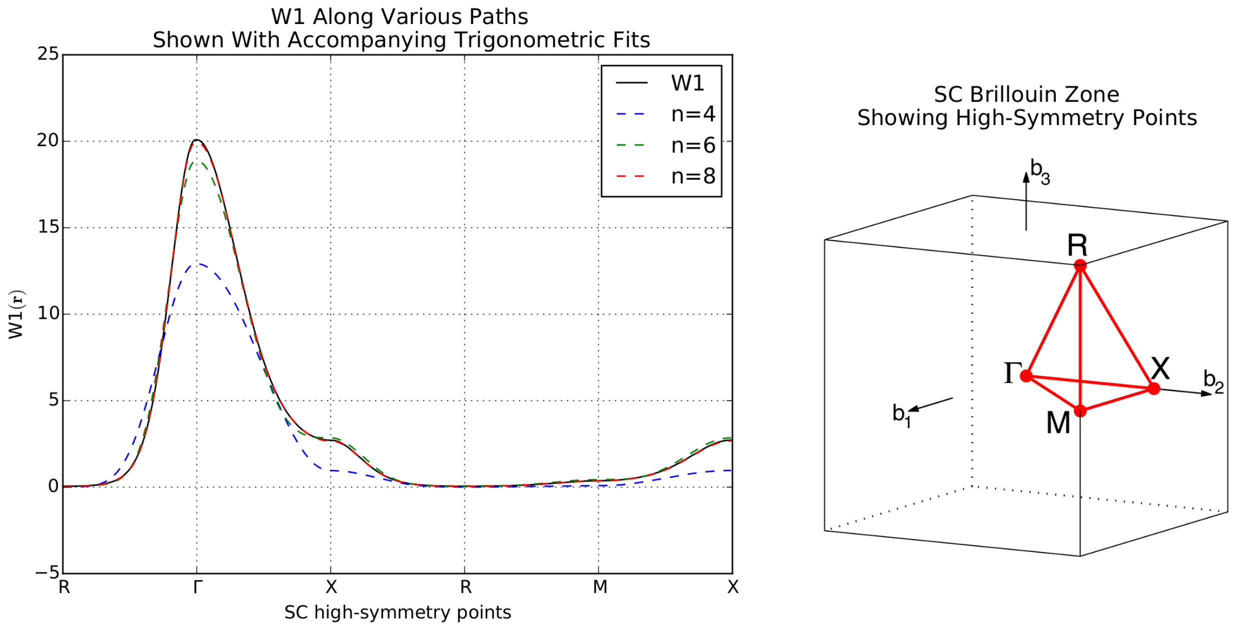

Figure 3.2 *Left:* High-symmetry paths of $W1$ and fits constructed with increasing numbers of sample points (n) in each dimension. *Right:* Corresponding Brillouin zone with the high-symmetry points labeled [12].

Though these outcomes are somewhat disconcerting, the trigonometric fits still converged to well below the target electron bands energy error threshold of 10^{-3} in both the reduced and unreduced cases. As such, we continued with further testing of the code's performance.

In addition to checking integral convergence, I plotted equation 3.1 and the fits made from various sample densities evaluated along certain high-symmetry lines in the unit cell, shown in Figure 3.2. This plot quite clearly shows the fit converging, with the $n = 7$ and $n = 8$ lines very difficult to distinguish from the actual values of the $W1$ function. These fits were all generated from the reduced sample data sets.

While further quantification is in order, figures 3.1 and 3.2 give quantitative and qualitative support indicating that our dftintegrate package is reasonably accurate in approximating our toy function.

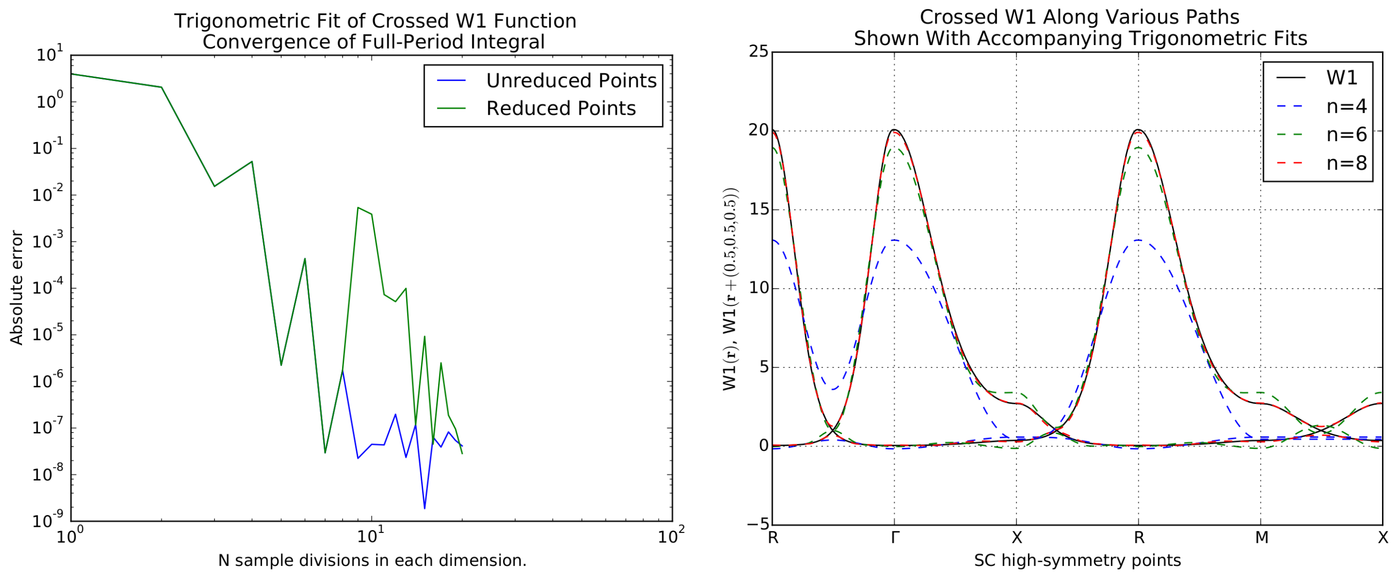

Figure 3.3 *Left:* Full-period integral convergence of trigonometric fits of crossed $W1$ functions generated with increasing sample points. *Right:* High-symmetry paths of crossed $W1$ functions and fits constructed with increasing numbers of sample points (n) in each dimension.

3.1.2 Multivalued, Smooth, Periodic Toy Function with Crossings

While the single valued $W1$ function is a useful test case to check that our interpolation method is properly implemented, it misses the typical complexity of the electron bands, which are multivalued functions. To test the behavior of dftintegrate on multivalued functions, I used the regular $W1$ function, as defined in equation 3.1 and a shifted version:

$$W1'(\mathbf{r}) = W1(\mathbf{r} + (0.5, 0.5, 0.5)). \tag{3.2}$$

For each sample point, the function returned values from both $W1$ and $W1'$ sorted from least to greatest. This simulates VASP data, where the data at each point is an ascending list of values. We fit all of the lower values with one set of coefficients, and then all the higher values with another set. This will introduce cusps at any location the functions cross, so each interpolation is fitting a non-smooth function.

We did not expect the trigonometric star-function basis to fit these crossed functions as well as the single-valued smooth function; however, we were hoping it would do well enough. The left of Figure 3.3 shows that the convergence of a full period integral, though not as smooth as in the single-valued case, still comes within 10^{-7} of the right answer. On the right, the actual function and several fits are shown evaluated along various paths.

While there are still concerns, these results are encouraging for using the dftintegrate module to represent actual DFT calculations. In the next section, however, I will show that in practice our implementation of trigonometric star-function interpolants did not do as well as we hoped on actual systems.

3.1.3 Aluminum Bands

When testing the trigonometric star-function basis on Aluminum (or any other real system) we do not have an analytic solution to the integral of the energy bands available, so I will not use convergence plots to quantify the effectiveness of the fit. To gain a qualitative idea of how well the fit is representing the electron bands, I present spaghetti plots along high-symmetry lines. Figure 3.4 shows actual band values—obtained by fine sampling in VASP—compared with trigonometric star-function interpolants of the lowest two bands built with symmetry-reduced sample grids of varying coarseness.

The interpolation of the first band looks reasonable along the paths involving Γ. However, it is quite apparent that the first band fit is not very good between points X, W, and L, and the interpolation of the second band does not look very good along any path. In the toy problem in the previous section, the trigonometric fit was able to handle the complexities introduced by the crossing. It appears, however, that the actual bands—being much more than bi-valued—have too many crossings and irregularities for the trigonometric basis to account for. The visual discrepancies between the actual data and the fits show that the trigonometric interpolation implemented in dftintegrate is

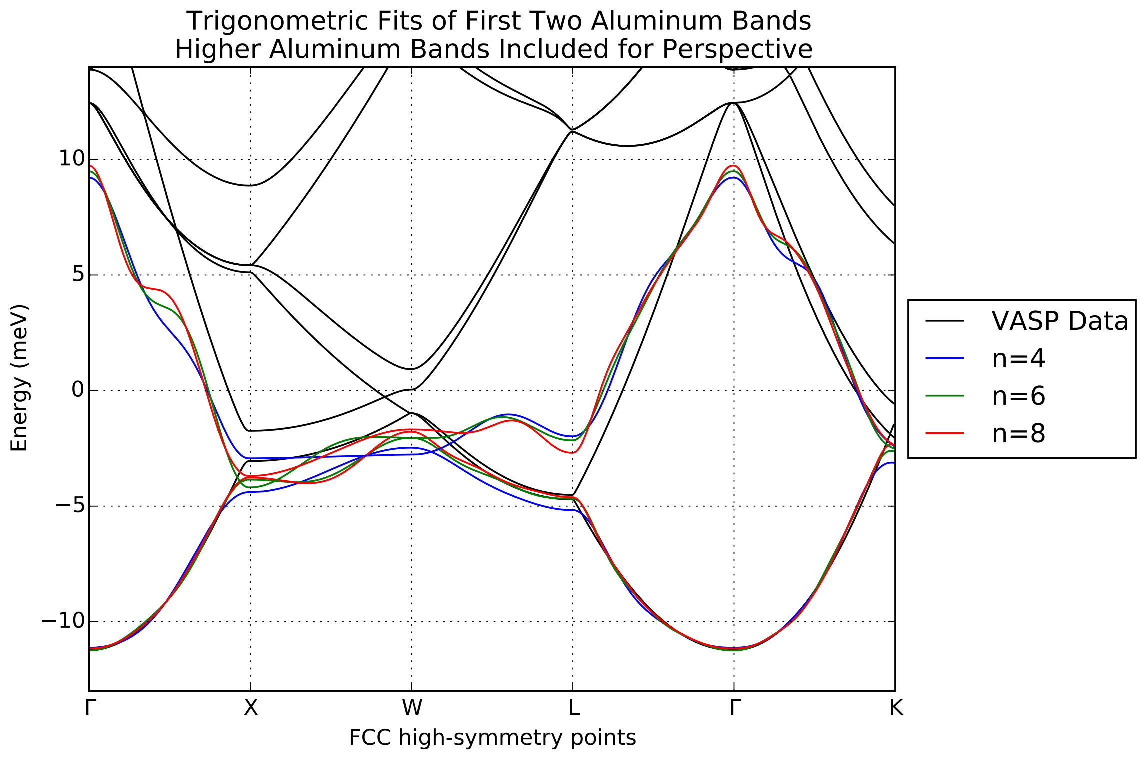

Figure 3.4 Aluminum band data sampled along high-symmetry lines is shown compared to various trigonometric fits. Each fit is constructed with a different symmetry-reduced sample, where n is the number of sample divisions in each dimension.

unfit for any of our production-line code without some way to account for the crossings. Attempts to address band crossings are address in section 3.3.

3.2 Periodic Spline Interpolation with ksi

In testing the implementation of the ksi Julia package, we started directly with spaghetti plots of fitting Aluminum bands, as pictured in Figure 3.5. This interpolation was made using a $24 \times 24 \times 24$ sample grid and 22^{nd}-degree polynomials (the highest degree allowed for this sample grid). The interpolation matches the first two bands extremely well, especially compared to trigonometric fits built using the same grid. On most paths, the fits of the third, fourth, and fifth bands do well.

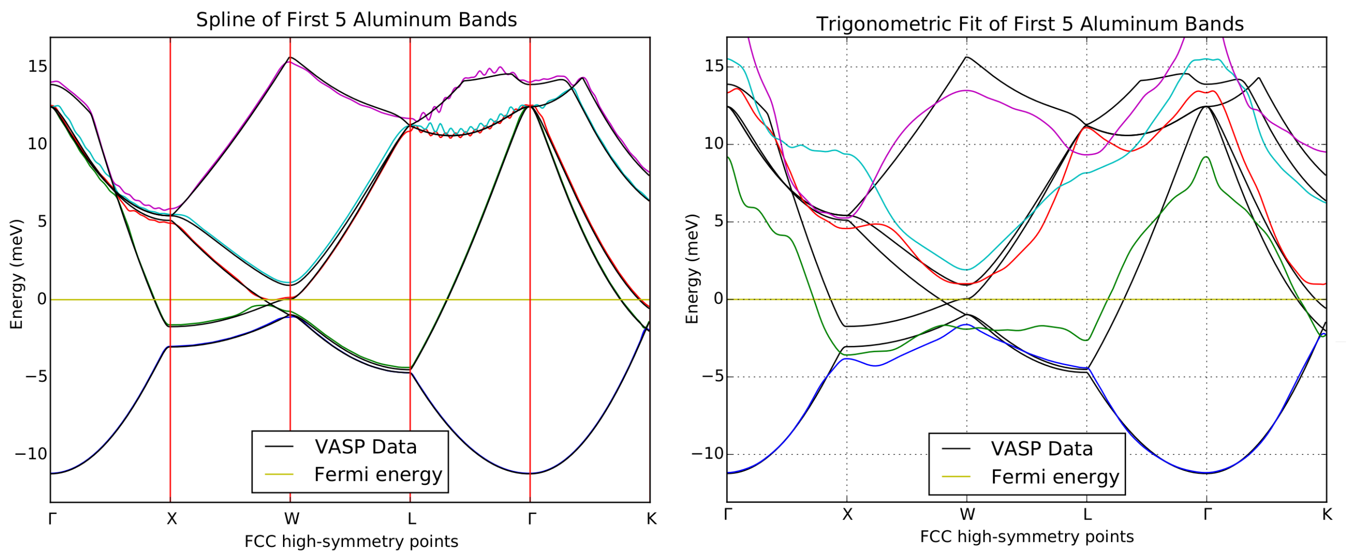

Figure 3.5 Data sampled ($N = 24$) along high-symmetry lines from the first five Aluminum bands are shown compared to spline interpolations (*left*) and trigonometric interpolations (*right*). Maximum degree ($d = 22$) polynomials were used in the spline interpolations (courtesy of Jeremy Jorgenson).

However, they show ringing on the Γ to X and Γ to L paths. Presumably, this is also due to cusps created when bands cross.

While the spline interpolation appears much more promising than the trigonometric interpolation, the ringing will potentially introduce greater inaccuracy in the band integration than we can tolerate. Addressing band crossings (section 3.3) will also be important before we can use the ksi package in production-line code.

3.3 Untangling Bands Attempts

As mentioned in section 1.3.1, separating the bands into individual, single-valued functions by sorting the energy values at each sample point from least to greatest is simple, but creates functions that are not very smooth. If we could separate the bands into individual functions with greater smoothness (like we can see on the spaghetti plots), our interpolation methods should be more effective. Below, I discuss the different methods we used to try and untangle the electron bands and effect our attempts had on the trigonometric interpolation.

3.3.1 SPD Decomposition

Our first attempt at untangling the bands used SPD decomposition. The *SPD* is in reference to S, P, and D atomic orbitals. One of the values VASP can calculate is the *projectability* of each energy value onto S, P, and D atomic orbitals. To visualize this, I created a spaghetti plot with varying colors to represent the S and P projectability at each sample point (Aluminum does not have any D electrons so there is no D projectability in this case). More cyan represents more S character, while more magenta represents more P character. More yellow represents a lower S and P projectability.

The SPD decomposition spaghetti plot generally shows smooth color gradients that match the smooth paths visually apparent in the original spaghetti plots (namely Figure 1.4). However, the SPD character still varies too much between different parts of the same band to make it useful when trying to untangle sparser sample sets. For example, the lowest band shows high S character near Γ, but between X, W, and L, several other bands that obviously are not smoothly connected to the lowest also show high S character, while the lowest band starts to exhibit much more P character in that region. The plot discouraged us from using SPD decomposition to attempt untangling the bands; if anything, it shows that SPD decomposition would lead to poor decisions when trying to untangle the bands.

3.3.2 Path Tracing Between Sample Points

We next tried a very manual approach to untangling the bands. We took a symmetry-reduced set of sample points and used VASP to sample finely in a straight path from one point to the next. Figure 3.7 demonstrates this for the 10 reduced sample points of a 4x4x4 grid. By tracing the paths visually from point to point, we determined which energy values likely belonged to the same

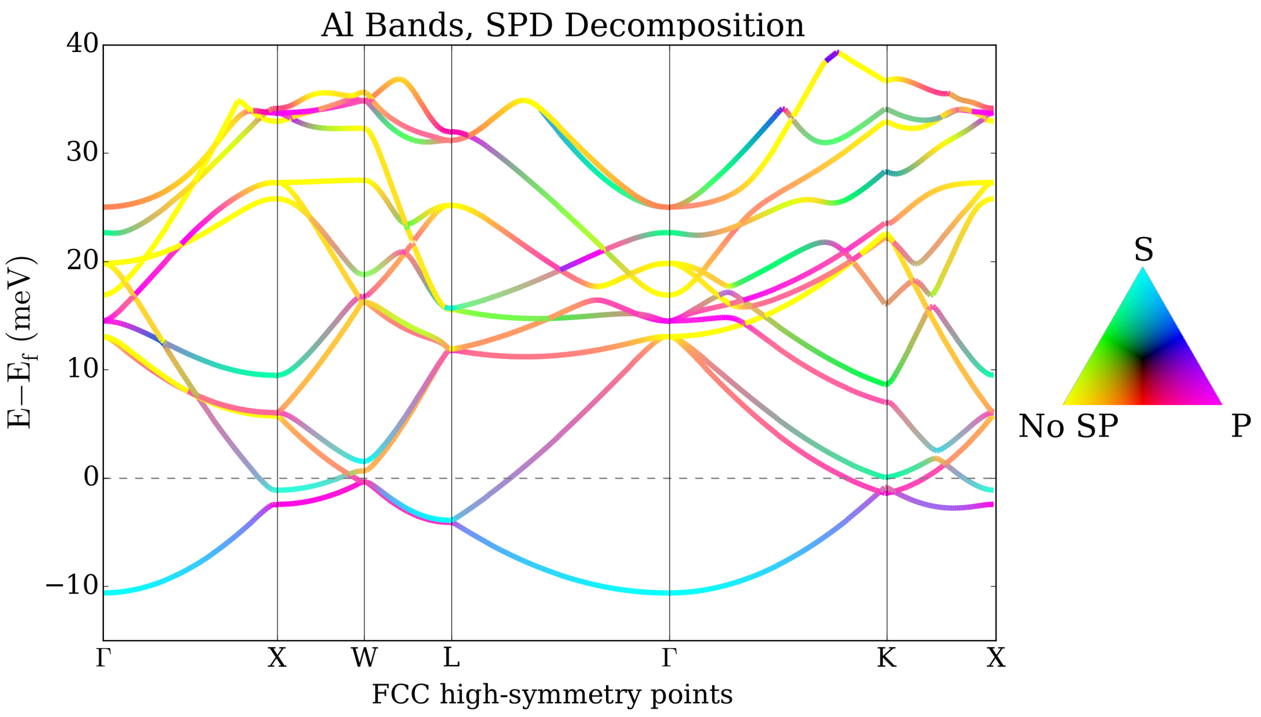

Figure 3.6 Aluminum band data sampled along high-symmetry lines is shown with the color representing the S and P character of each sample point. The S and P character depicted is relative to the maximum S and P character found in that band. For example, in the lowest band the greatest S character is at Γ, while the greatest P character at X, making the points cyan and magenta respectively.

band. For example, the second energy values at k-point 6 and 10 should probably be grouped with the lowest energy values for the other 8 sample points.

This approach was intended as a proof-of-concept that untangling the bands could help the interpolation methods. (Since it requires very fine sampling between points and manual grouping of energy values into individual bands, it is not a candidate for HT calculations even if it works.) Figure 3.8 shows the VASP data, original interpolations with untangling attempts, and new interpolations after attempting to untangle the bands. None of the uncrossing attempts show significant improvement over the original interpolations. While the uncrossed fits may better approach the actual band values near W, significant wiggling has been introduced near Γ. I tried many variations

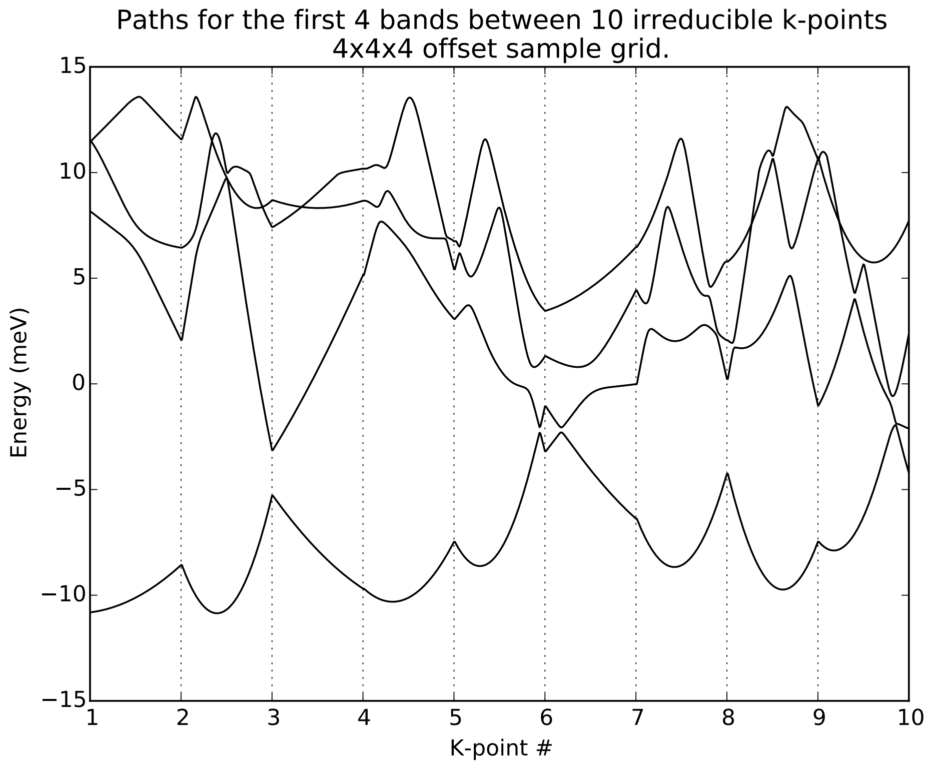

Figure 3.7 The electron bands were sampled densely between the 10 irreducible k-points of a 4x4x4 sample grid of aluminum. The paths indicate that the second value at k-points six and ten could be grouped with the lowest value at all other k-points to make a smooth first band.

and combination of the untangling for the $6 \times 6 \times 6$ and $8 \times 8 \times 8$ grids, none of which performed better than what is pictured here.

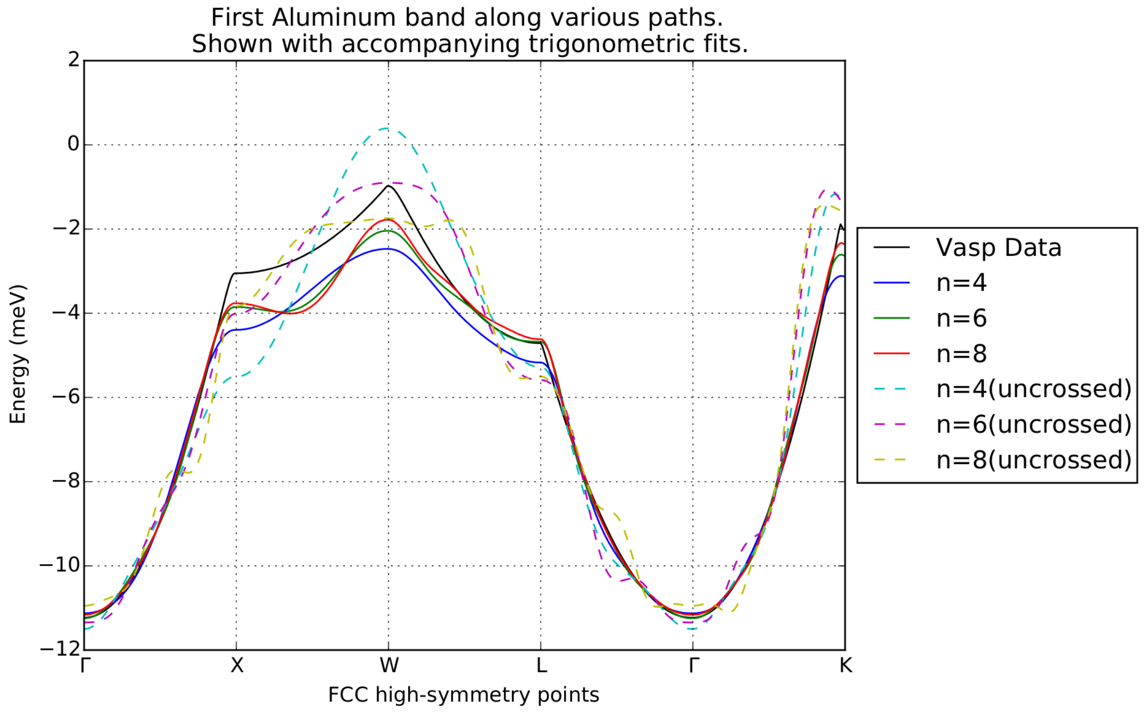

Figure 3.8 Densely sampled high-symmetry paths are compared to trigonometric interpolations generated using coarse sample data. The dashed lines are the interpolations after attempting to uncross the bands using path tracing.

Chapter 4

Conclusions and Future Work

Neither of the interpolation schemes we investigated was able to represent the complexity of the multivalued electron band functions in a satisfactory way. While the trigonometric interpolation approximated test functions well qualitatively (looking at the spaghetti plots) and quantitatively (considered the integral convergence), the large visual discrepancy between the VASP data and interpolation in Figure 3.4 destroys any hope of applying this method directly to DFT sample data. The spline interpolation shows more promise than the trigonometric interpolation, but still shows significant oscillations in Figure 3.5, likely due to the crossings and cusps inherent in the bands' multivalued nature. These oscillations will introduce more error into the total band energy than we can tolerate.

We tried to mitigate the issues caused by the multivalued nature of the bands by reordering the function values at each sample point. Our attempts to uncross the bands, both by explicitly sampling paths between sample points and by using SPD decomposition, were unsuccessful. We received no insights on reordering the bands (manually or automatically) from the SPD decomposition spaghetti plots. While it was not a feasible HT solution, tracing paths between sample points appeared to indicate which sample points had function values that needed to be reordered. However, Figure 3.8 gave no indication that reordering the function values made any improvements to the trigonometric fit.

Since none of our efforts in using an interpolation scheme to increase efficiency and accuracy have yielded favorable results, we are exploring new avenues. In particular, Agapito et. al. [15] have done work in Quantum Espresso (another DFT simulation program similar to VASP) that greatly reduces the size of the matrix used to calculate band energy sample points while still retaining accuracy up to a sufficiently high energy level. Their approach would allow us to more densely sample the bands while decreasing computation time. We are currently trying to integrate their method into our current HT codes for further testing.

Bibliography

[1] M. G. Initiative, "About the Materials Genome Initiative," https://www.mgi.gov/about (Accessed December 1, 2015).

[2] S. Curtarolo *et al.*, "AFLOW: An automatic framework for high-throughput materials discovery," Computational Materials Science **58,** 218–226 (2012).

[3] S. Curtarolo *et al.*, "AFLOWLIB.ORG: A distributed materials properties repository from high-throughput ab initio calculations," Computational Materials Science **58,** 227–235 (2012).

[4] C. for Material Genomics, "Automatic - FLOW for Materials Discovery," http://www.aflowlib.org/ (Accessed April 16, 2016).

[5] H. T. Stokes, *Solid State Physics: for Advanced Undergraduate Students*, 4 ed. (Brigham Young University, Provo, UT, 2007).

[6] "Bravais lattice," https://en.wikipedia.org/wiki/Bravais_lattice (Accessed December 5, 2015).

[7] J. William D. Callister, *Materials Science and Engineering: An Introduction*, 6 ed. (John Wiley & Sons, Inc., Hoboken, NJ, 2003), p. 68.

[8] J. William D. Callister, *Materials Science and Engineering: An Introduction*, 6 ed. (John Wiley & Sons, Inc., Hoboken, NJ, 2003), p. 274.

[9] U. W. Computational Materials Physics, “What is VASP?,” https://www.vasp.at/index.php/about-vasp/59-about-vasp (Accessed December 5, 2015).

[10] D. S. Sholl and J. A. Stechel, *Density Functional Theory: A Practical Introduction* (John Wiley & Sons, Inc., Hoboken, NJ, 2009).

[11] H. J. Monkhorst and J. D. Pack, “Special points for Brillouin-zone integrations,” Phys. Rev. B **13,** 5188–5192 (1976).

[12] W. Setyawan and S. Curtarolo, “High-throughput electronic band structure calculations: Challenges and tools,” Computational Materials Science **49,** 299–312 (2010).

[13] D. S. Durfee, *Physics phor Phynatics*, 1 ed. (Brigham Young University, Provo, UT, 2006), pp. 48–50.

[14] “NumPy v1.10 Manual,” http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.linalg.lstsq.html (Accessed January 9, 2016).

[15] L. A. Agapito, A. Ferretti, A. Calzolari, S. Curtarolo, and M. Buongiorno Nardelli, “Effective and accurate representation of extended Bloch states on finite Hilbert spaces,” Phys. Rev. B **88,** 165127 (2013).

Index