2019-04-01

Using Symmetry to Accelerate Materials Discovery

Wiley Spencer Morgan
Brigham Young University

Using Symmetry to Accelerate Materials Discovery

Wiley Spencer Morgan

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

Doctor of Philosophy

Department of Physics and Astronomy

Brigham Young University

ABSTRACT

Using Symmetry to Accelerate Materials Discovery

Wiley Spencer Morgan
Department of Physics and Astronomy, BYU
Doctor of Philosophy

Computational methods are commonly used by materials scientists to make predictions about materials. These methods can achieve in hours what would take days or weeks to accomplish in a lab. However, there are limits to what computational methods can do and how accurate the predictions are.

A limiting factor for computational materials science is the size of the search space. The space of potential materials is infinite. Selecting specific systems of elements on a fixed lattice to study reduces the number of possible arrangements of atoms in the lattice to a finite number. However, this number can still be very large. Additionally this list of arrangements will contain duplicates, i.e., two different atomic arrangements could be equivalent by a rotation or translation of the lattice. Using symmetry to eliminate the duplicates saves time and resources. In order to ensure that the final list of unique structures will fit into computer memory it is also useful to know how many unique arrangements there are before actually finding them. For this reason the Pòlya enumeration algorithm was created to determine the number of unique arrangements before enumerating them. A new atomic enumeration algorithm has also been implemented in the enumlib package. This new algorithm has been optimized to find the symmetrically unique arrangements for systems with large amounts of configurational freedom, such as high-entropy alloys, which have been too computationally expensive for other algorithms.

A popular computational method in materials science is Density Functional Theory (DFT). DFT codes perform first principles calculations by calculating the electron energy using numerical integrals. It is well known that the accuracy of the integrals depends heavily on the number of sample points, **k**-points, used. We have conducted a detailed study of how **k**-point sampling methods effect the accuracy of DFT calculations. This study shows that the most efficient **k**-point grids are those that have the fewest symmetrically distinct **k**-points, we call these *general regular* (GR) grids. GR grids are, however, difficult to generate, requiring a search across many possible grids. In order to make GR grids more accessible to the DFT community we have implemented an algorithm that can search **k**-point grids for the grid that has the fewest symmetry reduction in a matter of seconds.

Keywords: materials discovery, symmetry, numerical integration, sampling, Niggli, **k**-point folding

ACKNOWLEDGMENTS

It has been said that no man is an island. This document would not exist if it had not been for expert help and guidance along the way. Dr. Gus Hart has been an extraordinary advisor during my time here at BYU. He has introduced me to top rate researchers around the world and always helped me work toward my goals for post-graduate life.

Additionally many others here at BYU have made significant contributions to my work. There are a few I would like to make special note of; first, Dr. Rod Forcade's experience with algorithms, group theory, and math in general has been invaluable. Second, Dr. Branton Campbell has provided critical information and guidance on crystal lattice classification and manipulation. Finally I would like to acknowledge my peers in the "Materials Simulation Group" who have contributed code and ideas to the projects I have been a part of.

Finally I would like to thank my family, Sarah, Cora and River for being willing to be a part of this journey with me. I will always be grateful for the love and encouragement they have shown me.

USING SYMMETRY TO ACCELERATE

MATERIALS DISCOVERY

Wiley Spencer Morgan

Department of Physics and Astronomy
Brigham Young University
March 2019

Contents

List of Figures

CHAPTER 1

Symmetry in Materials Discovery

Materials have played a defining role in human achievement throughout history. The discovery of new materials in ages past has spurred the creation of the tools and technologies that created modern life. We are reaching the limits of what we can achieve with currently available materials. In order to continue to build new and exciting technologies new materials must be discovered.

1.1 QUANTUM MECHANICAL CALCULATIONS FOR MATERIALS DISCOVERY

Historically materials discovery has been a slow process that consists mainly of trial and error. Modern computational methods allow us to accelerate this process. What would take days or weeks to study in a lab can now be determined on supercomputers in hours. The accuracy of the computation depends on the modeling method used. More accurate models require more computational time and resources.

A popular computational method is Density Functional Theory [4, 5] (DFT) because it offers an excellent trade off between computation time and accuracy. DFT approximates solutions to the Schrödinger equation for a system of atoms at zero Kelvin. DFT's main drawback is that it can only compute the properties of one atomic configuration at a time. The number of possible binary materials, on a fixed lattice, is equivalent to the number of binary atomic combinations multiplied by the number of ways to place the atoms on the lattice. If the cell size of the binary systems is limited to 10 atoms then there are roughly 4 million possible materials to explore. Computing the properties of each set of binary materials using DFT would take a prohibitively long time even with today's supercomputers.

To overcome the difficulty presented by the size of the search space many materials scientists have started using high throughput [6–25] methods to construct large databases that can be searched for materials with desired properties. For example, as of January 2019 the AFLOW database contains data for 2.1 million crystal structures. Despite the many successes [26–36, 36–55] that have occurred using these databases they still represent a small portion of the possible materials space.

DFT calculations are also used to train machine learning models [56–58]. Machine learning models are statistical models that use sets of known data to make predictions about the properties of data outside of the training set. It is well known, however, that the accuracy of the learned model depends on the accuracy of the data used to train the model. Care must be taken when constructing the training data for these models. The advantage that machine learning offers is that once the model is trained it can make predictions that have close to DFT accuracy in a fraction of a second rather than hours. Using these methods researchers can quickly search over large sections of materials space for good material candidates.

Even with the aid of machine learning, materials scientists are still trying to search an infinite space using models that may have unreliable training data. These difficulties encourage the development of tools to 1) limit the materials search space, 2) ensure the accuracy of the training data being used, and 3) optimize the efficiency of individual computations.

1.2 REDUCING AN INFINITE SEARCH SPACE

There are a number of ways that the search space of materials can be limited. First, limiting the number of atoms in a crystal structure reduces the space to a countable, if large, number. Additional reductions can be achieved by: 1) selecting specific elements and lattices, 2) limiting the concentrations of each element, and 3) eliminating symmetrically equivalent crystals. Of these options, the first and second are easily accomplished, while the third, identifying symmetrically equivalent crystals, continues to give researchers difficulties, even today.

All crystal structures can be either translated or rotated by a symmetry of the lattice without changing the crystal's properties. Using symmetry to reduce a list of crystal structures to those that are unique then reduces the number of computations that need to be done to explore a materials system. Chapter 3 presents an algorithm that can efficiently list only the symmetrically distinct atomic arrangements for a crystal system.

It is also useful to know how many unique structures actually exist in order to ensure that the number of arrangements will fit into machine memory. This problem becomes increasingly important as the number of atoms and species in the cell increase. For example, a high-entropy alloy [59–62] (HEA) consisting of 5 atomic species of equal concentration within a 20 atom cell on a face-centered cubic (fcc) lattice will have around 10^9 unique arrangements of atoms, enough that the list alone becomes unwieldy. Chapter 2 presents a solution to this problem in the form of an algorithm that can determine the number of unique arrangements of atoms within a crystal structure without having to find or list them.

1.3 ERRORS FROM INTEGRATION SAMPLING

Data accuracy becomes increasingly important as databases of DFT calculations grow and are used in high throughput studies and to train machine learning models. In order to ensure that datasets are of a sufficient quality for these applications, the sources of errors must be understood and minimized to the extent possible. There are several known sources of error [63, 64] which can affect DFT calculations. **k**-point integration is one of these error sources that impacts all DFT calculations.

DFT calculations sample points in **k**-space to calculate electronic energy. It is well known that the calculation will be inaccurate if an insufficient number of **k**-points is used. Most studies start by performing convergence tests of the error with respect to the number of **k**-points to minimize errors. However, comprehensive studies of how **k**-point sampling affects the accuracy of calculations had never been studied until this work. Chapter 4 contains this detailed study of convergence rates of different types of **k**-point grids. This study also shows how difficult it is to be sure of the accuracy of a DFT calculation if insufficient sample points were used in a convergence test.

Chapter 4 also shows that the type of **k**-point grid used can make a profound difference on the accuracy and computational cost of a DFT calculation. Chapter 5 contains new algorithms that can quickly generate the most efficient grid.

1.4 CONCLUSION

The chapters in this dissertation contain tools and methods that can be used to accelerate materials discovery. The goals are to eliminate redundant calculations by only performing calculations for unique structures and by increasing the accuracy and efficiency of DFT calculations using algorithms that will generate optimal **k**-point grids.

While most of the algorithms that are described here have been implemented and are freely available to the scientific community, the algorithms in Chapter 5 have not yet been published. In implementing those algorithms, we have discovered that great care must be taken when handling finite precision or else roundoff error will cause failure. We are currently working on additional code that will prevent these failures. The main algorithms will not be released until the new code has been implemented so that the software package works reliably for the users.

The content of Chapters 2, 3, and 4 are centered around peer-reviewed articles, typeset in the style of the journal in which they were published. Each includes a description of the paper's context and future usage. The content of Chapter 5 contains the contents of another two papers each of which have been submitted for publication but not accepted yet.

CHAPTER 2

Sizing up the Search Space

As discussed in Section 1.2, the ability to enumerate every unique arrangement of atoms within a given crystal structure is an important aspect of computational materials discovery. This problem has been well studied and solved in general [1, 65–70]. For systems such as high entropy alloys (HEAs) however, the list of unique arrangements often exceeds the available computer memory. In such cases it is useful to know the number of unique arrangements *before* enumerating them.

The Pólya enumeration algorithm, found below, solves this problem. The algorithm is useful for the following reasons:

1. It allows researchers to determine if a system is too large or complex to be enumerated completely.

2. It allows alloy enumeration algorithms to check that enough memory exists to store the enumerated lists of atomic arrangements. The algorithm may then warn a user that the problem is too large and shut down rather than failing after hours or days of running.

3. It allows verification that alloy enumeration algorithms have found the correct number of unique structures.

The Pólya enumeration algorithm has been implemented in the `enumlib` package and is also available open source for both `FORTRAN` and `Python` at https://github.com/rosenbrockc/polya.

For this article, Conrad Rosenbrock wrote the original Python algorithm and the majority of the paper's text. I wrote the FORTRAN algorithm, implemented tests for both code implementations, and extended the algorithms to be able to handle an additional degree of freedom (displacement directions). The other authors helped edit the text and figures to make the algorithm easier to understand.

The following article is reproduced with permission. A license can be found in Appendix E.

Numerical Algorithm for Pólya Enumeration Theorem

CONRAD W. ROSENBROCK, WILEY S. MORGAN, and GUS L. W. HART,
Brigham Young University
STEFANO CURTAROLO, Duke University
RODNEY W. FORCADE, Brigham Young University

Although the Pólya enumeration theorem has been used extensively for decades, an optimized, purely numerical algorithm for calculating its coefficients is not readily available. We present such an algorithm for finding the number of unique colorings of a finite set under the action of a finite group.

Categories and Subject Descriptors: G.2.1 [**Discrete Mathematics**]: Combinatorics

General Terms: Combinatorial Algorithms, Counting Problems

Additional Key Words and Phrases: Pólya enumeration theorem, expansion coefficient, product of polynomials

ACM Reference Format:
Conrad W. Rosenbrock, Wiley S. Morgan, Gus L. W. Hart, Stefano Curtarolo, and Rodney W. Forcade. 2016. Numerical algorithm for pólya enumeration theorem. J. Exp. Algorithmics 21, 1, Article 1.11 (August 2016), 17 pages.
DOI: http://dx.doi.org/10.1145/2955094

1. INTRODUCTION

A circle partitioned into 4 equal sectors can be colored 16 different ways using two colors, $2^4 = 16$, as shown in Figure 1. But only 6 of these colorings are symmetrically distinct, several others being equivalent (under rotations and reflections) as shown by the arrows in the figure. The Pólya enumeration theorem provides a way to determine how many symmetrically distinct colorings there are with, for example, all sectors red (only one, as shown in the figure), one red sector and three green (again, only one), or the number with two red sectors and two green sectors (two, as shown in the figure). Borrowing a word from physics and chemistry, we refer to the partition of red and green sectors as the *stoichiometry*. For example, a coloring with 1 red sector and 3 green sectors has a stoichiometry of 1:3.

The Pólya theorem [Pólya 1937; Pólya and Read 1987] produces a polynomial (generating function), shown in the figure, whose coefficients answer the question of how many distinct colorings there are for each stoichiometry (each partition of the colors). For example, the $2r^2g^2$ term in the polynomial indicates that there are two distinct ways to color the circle with 2:2 stoichiometry (⊗⊗). For all other stoichiometries (4:0,

This work was supported under Grant No. ONR (MURI N00014-13-1-0635).
Authors' addresses: C. W. Rosenbrock, W. S. Morgan, and G. L. W. Hart, Department of Physics and Astronomy, 84602, Brigham Young University; S. Curtarolo, Materials Science, Electrical Engineering, Physics and Chemistry, 27708, Duke University; R. W. Forcade, Department of Mathematics, 84602, Brigham Young University.

$$P(r, g) = r^4 + r^3 g + 2r^2 g^2 + r g^3 + g^4$$

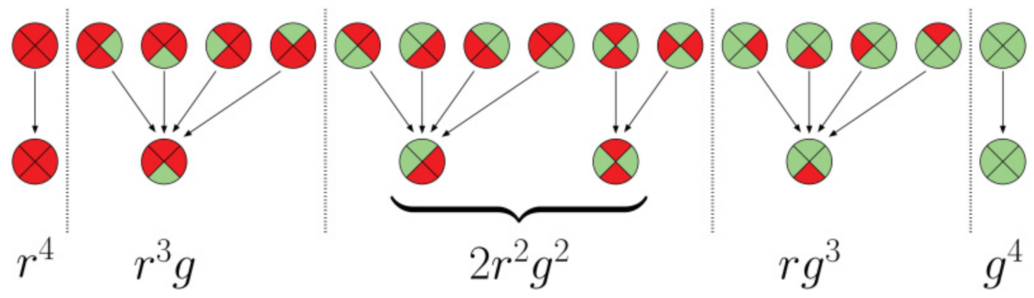


Fig. 1. Top row: All possible two-color colorings of a circle divided into four equal sectors (left side of figure). Bottom row: All symmetrically distinct binary colorings of the circle. Arrows indicate combinatorically distinct colorings that are equivalent by symmetry.

0:4, 1:3, and 3:1), the polynomial coefficients are all 1, indicating that for each of these cases there is only one distinct coloring, as is obvious from the figure.

A common problem in many fields involves enumerating[1] the *symmetrically distinct* colorings of a finite set, similar to the toy problem of Figure 1. The Pólya theorem has shown its wide range of applications in a variety of contexts. Classically, it was applied to counting chemical isomers [Robinson et al. 1976; Kennedy et al. 1964; Pólya 1937] and graphs [Harary 1955]. Recent examples include confirming enumerations of molecules in bioinformatics and chemoinformatics [Deng and Qian 2014; Ghorbani and Songhori 2014]; unlabeled, uniform hypergraphs in discrete mathematics [Qian 2014]; analysis of tone rows in musical composition [Lackner et al. 2015]; commutative binary models of Boolean functions in computer science [Genitrini et al. 2015]; generating functions for single-trace-operators in high-energy physics [McGrane et al. 2015]; investigating the role of nonlocality in quantum many-body systems [Tura et al. 2015]; and photosensitizers in photosynthesis research [Taniguchi et al. 2014].

In computational materials science, chemistry, and related subfields such as computational drug discovery, combinatorial searches are becoming increasingly important, especially in high-throughput studies [Curtarolo et al. 2013]. As computational methods gain a larger market share in materials and drug discovery, algorithms such as the one presented in this article are important as they provide validation support to complex enumeration codes. Pólya's theorem is the only way to independently confirm that an enumeration algorithm has performed correctly. The present algorithm has been useful in checking a new algorithm extending the work in Hart and Forcade [2008, 2009] and Hart et al. [2012], and Pólya's theorem was recently used in a similar crystal enumeration algorithm [Mustapha et al. 2013] that has been incorporated into the CRYSTAL14 software package [Dovesi et al. 2014].

Despite the widespread use of Pólya's theorem in different science and mathematics contexts, a low-level, numerical implementation is not available. Typical approaches use Computer Algebra Systems (CASs) to symbolically generate the Pólya polynomial. This strategy is ineffective for two reasons. First, CASs are too slow for large problems that arise in a research setting, and, second, generating the entire Pólya polynomial (which can have billions or trillions of terms) is unnecessary when one is interested in only a single stoichiometry.

[1]The Pólya theorem does not generate the list of unique colorings (which is generally a much harder problem), but it does determine the *number* of unique colorings.

Here we demonstrate a low-level algorithm for finding the polynomial coefficient corresponding to a single stoichiometry. It exploits the properties of polynomials and *a priori* knowledge of the relevant term. We briefly describe the Pólya enumeration theorem in Section 2, followed by the algorithm for calculating the polynomial coefficients in Section 3. In the final section, we investigate the scaling and performance of the algorithm.

2. PÓLYA ENUMERATION THEOREM

2.1. Introduction the Pólya Enumeration Theorem

Pólya's theorem provides a simple way to construct a generating polynomial whose coefficients count the numbers of symmetrically distinct colorings for each possible stoichiometry. The polynomial in Figure 1 above was easy to verify because we were able to hand count the symmetrically distinct colorings. But suppose there were dozens of colors and dozens of sites to be colored and hundreds of symmetries to apply. In that case, it is easier to use Pólya's theorem to construct the polynomial directly from the symmetry group.

To describe this very useful theorem, we refer once more to Figure 1. There are four symmetries—the identity, two 90° rotations (clockwise and counterclockwise), and a 180° rotation. If we label the colorable sectors 1, 2, 3, and 4, and write the permutations in *disjoint-cycle* notation, we have (1)(2)(3)(4) for the identity, the two 90° rotations are represented by (1234) and (1432), while the 180° rotation is (13)(24) in cycle notation.

Now Pólya's theorem simply tells us to replace each cycle of length λ with a sum of λ-th powers of variables corresponding to the colors available. For example, letting r and g stand for red and green, the identity is represented by $(r+g)(r+g)(r+g)(r+g)$, the two 90° rotations are each replaced by (r^4+g^4), and the 180° rotation is replaced by $(r^2+g^2)(r^2+g^2)$. When we *average* these four polynomials, we get the Pólya polynomial predicted above:

$$\begin{aligned} P(r,g) &= \frac{1}{4}\big((r+g)(r+g)(r+g)(r+g) + (r^4+g^4) + (r^4+g^4) + (r^2+g^2)(r^2+g^2)\big) \\ &= r^4 + r^3g + 2r^2g^2 + rg^3 + g^4. \end{aligned} \tag{1}$$

In other words, Pólya's theorem relies on a structural representation of the symmetries *as permutations written in disjoint-cycle notation* to construct the generating polynomial we need.

The problem with Pólya, however, is that it requires us to compute the *entire* polynomial when we may need only one of its coefficients. For example, if we have 50 sites to color, and 20 colors available, the number of *terms* in our polynomial (regardless of symmetries) would be about 4.6×10^{16}. That is a lot of work (and memory) to compute the entire polynomial (and all of those very large terms) if we needed *only* to know the number of symmetrically distinct colorings for a single stoichiometry. That information is contained in just 1 term of the 46 quadrillion terms of the Pólya polynomial. Can we spare ourselves the work of computing all the others?

Suppose we have a target stoichiometry $[c_1 : c_2 : \cdots : c_\xi]$, where ξ is the number of colors and $\sum_{j=1}^{\xi} c_j = n$ is the number of sites to be colored. To find the number of symmetrically distinct colorings with those frequencies, we must determine the coefficient of the single term in the Pólya polynomial containing the product $x_1^{c_1} x_2^{c_2} \ldots x_\xi^{c_\xi}$. The Pólya polynomial is the average,

$$P(x_1, x_2, \ldots, x_\xi) = \frac{1}{|G|}\left(\sum_{\pi \in G} P_\pi(x_1, x_2, \ldots, x_\xi)\right), \tag{2}$$

of the polynomials $P_\pi(x_1, x_2, \dots, x_\xi)$ computed for each permutation π in the symmetry group G, each P_π being formed by multiplying the representations of each disjoint cycle in π (as illustrated in Equation (1)).

Clearly, if we are only interested in the coefficient of $x_1^{c_1} x_2^{c_2} \dots x_\xi^{c_\xi}$ in P, we may simply find the coefficient of that product in each P_π and add those partial coefficients together. Thus, given a permutation π with k_1 cycles of length r_1, k_2 cycles of length r_2, and so on, up to k_t cycles of length r_t, with $\sum_{i=1}^{t} r_i k_i = n$ (the number of sites, t is the number of cycle types), we must compute the coefficient of $x_1^{c_1} x_2^{c_2} \dots x_\xi^{c_\xi}$ in P_π.

It is well known that a product of sums is equal to the sum of all products one can obtain by taking one summand from each factor (generalizing the familiar First Outer Inner Last (FOIL) rule used by undergrads to multiply two binomials). Thus the polynomial P_π is the sum of all products of the form $\prod_s x_{i_s}^{\lambda(s)}$ (where the product runs over all cycles s, $\lambda(s)$ is the length of the cycle s, and x_{i_s} is one of the colors chosen from the sum for that cycle). Thus the product we want, $x_1^{c_1} x_2^{c_2} \dots x_\xi^{c_\xi}$, has a coefficient that simply counts the number of products of the form $\prod_s x_{i_s}^{\lambda(s)}$ where the sum of the exponents for each x_i is equal to the target c_i.

Each cycle, of length r_i ($i = 1 \dots t$), gets assigned to one of the colors. Let s_{ij} be the number of cycles of length r_i assigned to color j ($j = 1 \dots \xi$). This defines a $t \times \xi$ matrix $S = (s_{ij})$ of non-negative integers, where (1) the sum of row i equals the number of cycles of length r_i:

$$\sum_{j=1}^{\xi} s_{ij} = k_i \quad \text{(row sum condition)}, \tag{3}$$

and (2) weighted sum of column j must equal the target frequency of the j-th color:

$$\sum_{i=1}^{t} r_i s_{ij} = c_j \quad \text{(column sum condition)}, \tag{4}$$

in order to achieve our target stoichiometry.

For each such matrix, there are a number of possible ways to assign colors to the cycles, with multiplicities prescribed by S. The number is

$$F(S) = \prod_{i=1}^{t} \binom{k_i}{s_{i1}, s_{i2}, \dots, s_{i\xi}}, \tag{5}$$

the product of the number of ways to do it for each cycle. Thus we are obliged to sum the function $F(S)$, so computed, over all matrices S meeting the given row and column sum conditions (3) and (4).

If we do this computation for each permutation π, and average them (add them and divide by $|G|$), we then get the coefficient of the Pólya polynomial $P(x_1, x_2, \dots, x_i)$ corresponding to our target stoichiometry $[c_1 : c_2 : \cdots : c_\xi]$. This calculation depends only on the *cycle type* of the permutation, the number of disjoint cycles of different lengths comprising the disjoint-cycle representation. Thus we only need to make an inventory of the cycle types for our permutations and do the calculation once for each distinct cycle type. There will not be more such cycle types than the number of conjugacy classes in the symmetry group. Also, note, the utility of multinomial coefficients in this context stems from the likelihood that our permutations will have many cycles of the same length.

Algorithmically, the process is straight forward. First, we must find all matrices S which meet the row and sum conditions (3) and (4) above. For each successful matrix,

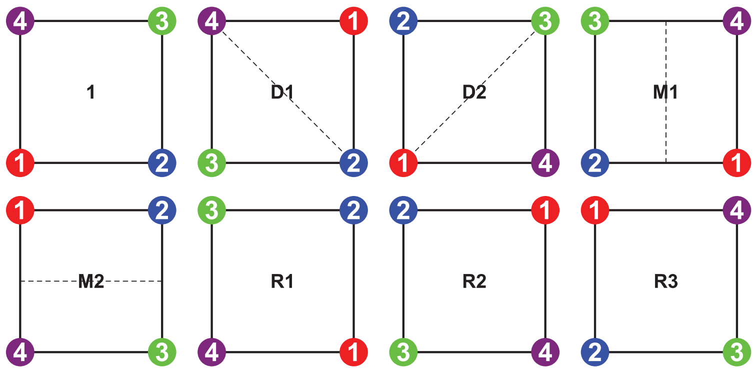


Fig. 2. The symmetry group operations of the square. This group is known as the dihedral group of degree 4 or D_4. The dashed lines are guides to the eye for the horizontal, vertical, and diagonal reflections (**M1**,**M2** and **D1**, **D2**).

we then compute the product of row-multinomial-coefficients. We add those up and multiply by the number of permutations in the conjugacy class, sum those results for the conjugacy classes, and divide by the group order. That gives us the Pólya coefficient for the given stoichiometry.

For example, suppose our permutation is made up of two 1-cycles, three 2-cycles, and one 4-cycle (so the number of sites is 12), and we have three colors with frequencies (red:green:blue $\rightarrow$ 4:6:2) respectively. Then we are looking for 3×3 matrices S whose rows sum to $\begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$ and whose columns (when dotted with the cycle lengths $\begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$) sum to 4, 6, and 2 respectively. There are exactly five such matrices (see Figure 3 and discussion in Section 3):

$$\begin{pmatrix} 0 & 0 & 2 \\ 0 & 3 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 2 \\ 2 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 2 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{6}$$

The multinomial coefficient for the top and bottom row in each case is $\binom{2}{2,0,0} = \binom{2}{2} = 1 = \binom{1}{1,0,0}$, so the $F(S)$ in each case is equal to the multinomial coefficient of the middle row; thus $\binom{3}{3} = 1$ in the first case, $\binom{3}{2,1} = 3$ for the middle three matrices, and $\binom{3}{1,1,1} = 6$ for the right-hand matrix. So our count for this problem is $1 + 3 + 3 + 3 + 6 = 16$. We may check this by computing $(r + g + b)^2(r^2 + g^2 + b^2)^3(r^4 + g^4 + b^4)$ (a la Pólya) and noting that the coefficient of $r^4g^6b^2$ is indeed 16.

Clearly, we can do that for each permutation in the group and sum the results. That is equivalent to determining in how many ways we may assign a single color to each cycle in the permutation—in such a way that the total number of occurrences of each color achieves its target frequency.

2.2. Example: Applying Pólyas Theorem

Here we present a simple example showing how Pólya's theorem is applied to a small, finite group. The square has the set of symmetries displayed in Figure 2. These symmetries include four rotations (by 0°, 90°, 180°, and 270°; labeled **1**, **R1**, **R2**, and **R3**) and four reflections (one horizontal, one vertical, and two for the diagonals; labeled

Table I. Disjoint-Cyclic Form for Each Group Operation in D_4 and the Corresponding Polynomials, Expanded Polynomials and the Coefficient of the x^2y^2 Term for Each

Op.	Disjoint-Cyclic	Polynomial	Expanded	Coeff.
$\mathbb{1}$	(1)(2)(3)(4)	$(x+y)^4$	$x^4+4x^3y+\boxed{6x^2y^2}+4xy^3+y^4$	6
D1	$(1,3)(2)(4)$	$(x^2+y^2)(x+y)^2$	$x^4+2x^3y+\boxed{2x^2y^2}+2xy^3+y^4$	2
D2	$(1)(2,4)(3)$	$(x^2+y^2)(x+y)^2$	$x^4+2x^3y+\boxed{2x^2y^2}+2xy^3+y^4$	2
M1	$(1,2)(3,4)$	$(x^2+y^2)^2$	$x^4+\boxed{2x^2y^2}+y^4$	2
M2	$(1,4)(2,3)$	$(x^2+y^2)^2$	$x^4+\boxed{2x^2y^2}+y^4$	2
R1	$(1,4,3,2)$	(x^4+y^4)	$x^4+\quad+y^4$	0
R2	$(1,3)(2,4)$	$(x^2+y^2)^2$	$x^4+\boxed{2x^2y^2}+y^4$	2
R3	$(1,2,3,4)$	(x^4+y^4)	$x^4+\quad+y^4$	0

M1, **M2** and **D1**, **D2**). This group is commonly known as the dihedral group of degree four, or D_4 for short.[2]

The group operations of the D_4 group can be written in disjoint-cyclic form as in Table I. For each r-cycle in the group, we can write a polynomial in variables x_i^r for $i = 1 \dots \xi$, where ξ is the number of colors used. For this example, we will consider the situation where we want to color the four corners of the square with only two colors. In that case we end up with just two variables x_1, x_2, which are represented as x, y in the table.

The Pólya representation for a single group operation in disjoint-cyclic form results in a product of polynomials that we can expand. For example, the group operation **D1** has disjoint-cyclic form $(1,3)(2)(4)$ that can be represented by the polynomial $(x^2 + y^2)(x + y)(x + y)$, where the exponent on each variable corresponds to the length of the r-cycle of which it is a part. For a general r-cycle, the polynomial takes the form

$$\left(x_1^r + x_2^r + \cdots + x_\xi^r\right), \tag{7}$$

for an enumeration with ξ colors. As described in Section 2.1, we exchange the group operations acting on the set for polynomial representations that obey the familiar rules for polynomials.

We will now pursue our example of the possible colorings on the four corners of the square involving two of each color. Excluding the symmetry operations, we could come up with $\binom{4}{2} = 6$ possibilities, but some of these are equivalent by symmetry. The Pólya theorem counts how many *unique* colorings we should recover. To find that number, we look at the coefficient of the term corresponding to the overall color selection (in this example, two of each color); thus we look for coefficients of the x^2y^2 term for each group operation. These coefficient values are listed in Table I. The sum of these coefficients, divided by the number of operations in the group, gives the total number of unique colorings under the entire group action, in this case $(6 + 2 + 2 + 2 + 2 + 0 + 2 + 0)/8 = 16/8 = 2$.

Next, we apply the procedure discussed in connection with Equation (6) to construct the matrix S for one of the permutations of the square. It illustrates the idea behind the general algorithm presented in the next section.

In the symmetries of the square, there is a cycle type consisting of two 1-cycles and one 2-cycle. The two permutations with that type are (1)(3)(24) and (2)(4)(13). The cycle lengths are 1 (with multiplicity 2) and 2 (with multiplicity 1). So each of those

[2]The dihedral groups have multiple, equivalent names. D_4 is also called Dih_4 or the dihedral group of *order* 8 (D_8).

permutations requires a matrix $S = \begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix}$ satisfying $s_{11} + s_{12} = 2$ and $s_{21} + s_{22} = 1$ (row sum condition (3)) and $s_{11} + 2s_{21} = 2$ and $s_{12} + 2s_{22} = 2$ (column sum condition (4)). There are only two matrices of non-negative integers satisfying those conditions simultaneously:

$$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}. \tag{8}$$

For each of these matrices, the row-multinomial coefficients are $\binom{2}{0,2} = 1$ and $\binom{1}{0,1} = 1$ so each matrix yields a product 1. Thus each permutation of this cycle type contributes 2 to the sum. This corresponds to the fact that the coefficient of x^2y^2 in $(x+y)^2(x^2+y^2)$ is 2.

Since there are two permutations of this cycle type, the total contribution of the cycle type to the overall Pólya polynomial is 4 (which must then be divided by the number of symmetries in the group).

Thus, in general, the only problem is to find an efficient way of generating these matrix solutions. Since the problem is equivalent to enumerating all lattice points within a high-dimensional polytope, we presume that a tree search (implemented recursively or via a backtracking algorithm) may be the most efficient way to achieve this.

3. COEFFICIENT-FINDING ALGORITHM

Our implementation of the tree search is fundamentally identical to the method of the last section; however, the details may not be immediately recognizable as such.[3] In this section we rephrase the row and column sum conditions (3) and (4) to highlight the logical connections between our specific implementation and the general ideas from Section 2. We adopt this approach because (1) for pedagogical value, the matrix approach is much easier to visualize and (2) the algorithms presented here mirror the accompanying code closely, which we consider valuable.

First, for a generic polynomial

$$\left(x_1^r + x_2^r + \cdots + x_\xi^r\right)^d, \tag{9}$$

the exponents of each x_i in the *expanded* polynomial are constrained to the set

$$V = \{0, r, 2r, 3r, \ldots, dr\}. \tag{10}$$

Next, we consider the terms in the expansion of the polynomial:

$$\left(x_1^r + x_2^r + \cdots + x_\xi^r\right)^d = \sum_{k_1,k_2,\ldots,k_\xi} \mu_k \prod_{i=1}^{\xi} x_i^{rk_i}, \tag{11}$$

where the sum is over all possibles sequences $k_1, k_2, \ldots, k_\xi$ such that the sum of the exponents (represented by the sequence in k_i) is equal to d,

$$k_1 + k_2 + \cdots + k_\xi = d. \tag{12}$$

[3]If all you are looking for is a working code, you now know enough to use it. Download it at https://github.com/rosenbrockc/polya.

As described in the introduction, the coefficients μ_k in the polynomial expansion Equation (11) are found using the multinomial coefficients

$$\begin{aligned}\mu_k &= \binom{n}{k_1, k_2, \ldots, k_\xi} = \frac{n!}{k_1! k_2! \cdots k_\xi!} \\ &= \binom{k_1}{k_1}\binom{k_1+k_2}{k_2}\cdots\binom{k_1+k_2+\cdots+k_\xi}{k_\xi} \\ &= \prod_{i=1}^{\xi}\binom{\sum_{j=1}^{i} k_j}{k_i}. \end{aligned} \tag{13}$$

Finally, we define the polynomial (7) for an arbitrary group operation $\pi \in \mathbf{G}$ as[4]

$$P_\pi(x_1, x_2, \ldots, x_\xi) = \prod_{\alpha=1}^{m} M_\alpha^{r_\alpha}(x_1, x_2, \ldots, x_\xi), \tag{14}$$

where each $M_\alpha^{r_\alpha}$ is a polynomial of the form (9) for the α^{th} distinct r-cycle and d_α is the multiplicity of that r-cycle; m is the number of cycle types in P_π. Linking back to the matrix formulation, each $M_\alpha^{r_\alpha}$ is equivalent to a *row* S_i in matrix S.

Since we know the fixed stoichiometry term $T = \prod_{i=1}^{\xi} T_i = \prod_{i=1}^{\xi} x_i^{c_i}$ in advance, we can limit the possible sequences of k_i for which multinomial coefficients are calculated. This is the key idea of the algorithm and the reason for its high performance.

For each group operation π, we have a product of polynomials $M_\alpha^{r_\alpha}$. We begin filtering the sequences by choosing only those combinations of values $v_{i\alpha} \in V_\alpha = \{v_{i\alpha}\}_{i=1}^{d_\alpha+1}$ for which the sum

$$\sum_{\alpha=1}^{m} v_{i\alpha} = T_i, \tag{15}$$

where V_α is the set from Eq. (10) for multinomial $M_\alpha^{r_\alpha}$. At this point it is useful to refer to Figure 3 to make the connection to the recursive tree search for possible matrices. The V_α are equivalent to all the possible values that any of the elements in a row of the matrix may take. If we take $M_1^{r_1}$ as an example, then V_1 is the collection of all values that show up in row 1 of any matrix in the figure, multiplied by the number of cycles with length r_1. Constraint (15) is equivalent to the column sum requirement (4).

We first apply constraint (15) to the x_1 term across the product of polynomials to find a set of values $\{k_{1\alpha}\}_{\alpha=1}^{m}$ that could give exponent T_1 once all the polynomials' terms have been expanded. This is equivalent to finding the set of first columns in each matrix that match the target frequency for the first color. Once a value $k_{1\alpha}$ has been fixed for each $M_\alpha^{r_\alpha}$, the remaining exponents in the sequence $\{k_{1\alpha}\} \cup \{k_{i\alpha}\}_{i=2}^{\xi}$ are constrained via (12). We can recursively examine each variable x_i in turn using these constraints to build a set of sequences

$$S_l = \{S_{l\alpha}\}_{\alpha=1}^{m} = \{(k_{1\alpha}, k_{2\alpha}, \ldots, k_{\xi\alpha})\}_{\alpha=1}^{m}, \tag{16}$$

where each $S_{l\alpha}$ defines the exponent sequence for its polynomial $M_\alpha^{r_\alpha}$ that will produce the target term T after the product is expanded. Each $S_{l\alpha} \in S_l$ represents the transposed matrix S that survives both the row and column sum conditions (highlighted in green in the figure). Thus, S_l is the set of these matrices for the group operation π. The

[4]We will use Greek subscripts to label the polynomials in the product and Latin subscripts to label the variables within any of the polynomials.

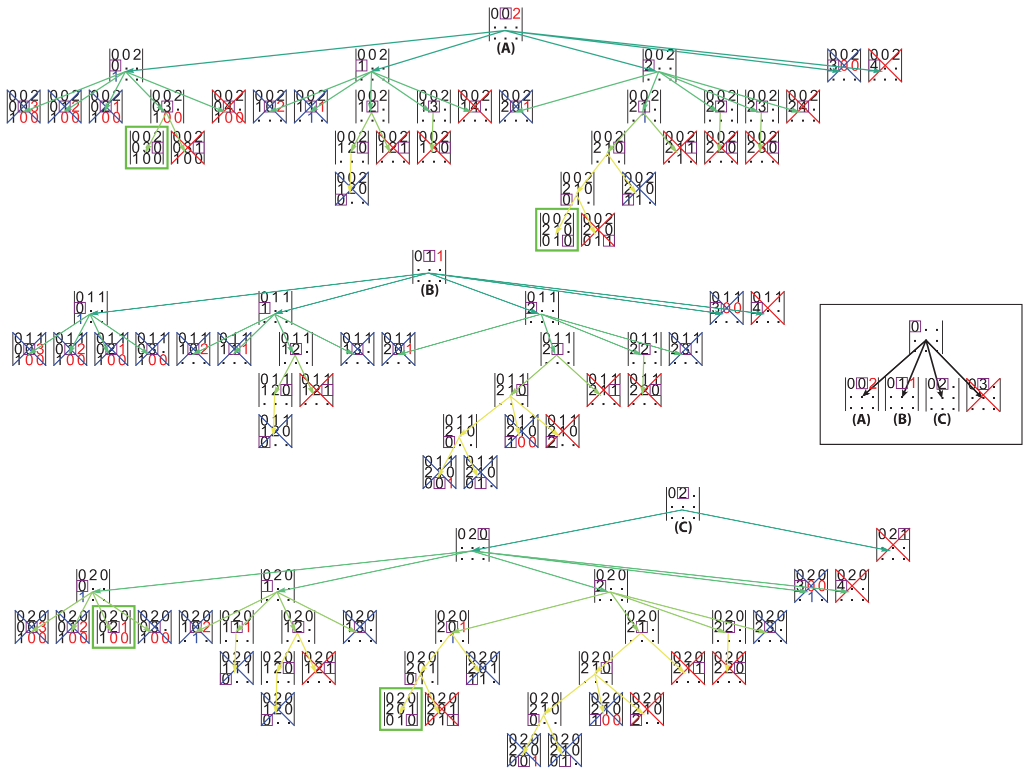


Fig. 3. A recursive tree search for some of the possible matrices S for the problem of Section 2: two 1-cycles, three 2-cycles, and one 4-cycle. We have restricted the figure to include only the zero pendants of the tree, which produce four of the five relevant matrices in Equation (6). Matrix elements in red (blue) represent the only possible values that would satisfy the row (column) sum conditions. A red (blue) cross over a matrix shows that it fails the row (column) sum condition, and its descendants need not be examined. Matrices with green borders are solutions to the tree search problem. The purple squares show the current row and column on which the recursive search is operating.

maximum value of l depends on the target term T and how many possible $v_{i\alpha}$ values are filtered out using constraints (15) and (12) at each step in the recursion.

Once the set $\mathbf{S} = \{S_l\}$ has been constructed, we use Equation (13) on each polynomial's $\{k_{i\alpha}\}_{i=1}^{\xi}$ in $S_{l\alpha}$ to find the contributing coefficients. The final coefficient value for term T resulting from group operation π is

$$t_\pi = \sum_l \tau_l = \sum_l \prod_{\alpha=1}^{m} \binom{d_\alpha}{S_{l\alpha}}. \tag{17}$$

To find the total number of unique colorings under the group action, this process is applied to each element $\pi \in \mathbf{G}$ and the results are summed and then divided by $|\mathbf{G}|$.

We can further optimize the search for contributing terms by ordering the exponents in the target term T in descending order. All the $\{k_{1\alpha}\}_{\alpha=1}^{m}$ need to sum to T_1 (15); larger values for T_1 are more likely to result in smaller sets of $\{k_{i\alpha}\}_{\alpha=1}^{m}$ across the polynomials. This happens because if T_1 has smaller values (like 1 or 2), then we end up with lots of possible ways to arrange them to sum to T_1 (which is not the the case for the larger values). Since the final set of sequences S_l is formed using a Cartesian product, including a few extra sequences from any T_i prunings multiplies the total number of sequences significantly. In the figure, this optimization is equivalent to completing a row with red entries because all the remaining, unfilled entries are constrained by the row sum condition.

Additionally, constraint (12) applied within each polynomial will also reduce the total number of sequences to consider if the first variables x_1, x_2, and so on, are larger integers compared to the target values T_1, T_2, and so on. This speed-up comes from the recursive implementation: If x_1 is already too large (compared to T_1), then possible values for $x_2, x_3, \dots$ are never considered. This optimization is equivalent to completing matrix columns with blue entries because of the column sum constraint.

3.1. Pseudocode Implementation

Note: Implementations in PYTHON and FORTRAN are available in the supplementary material.

For both algorithms presented below, the operator ($\Leftarrow$) pushes the value to its right onto the list to its left.

For algorithm (1) in the EXPAND procedure, the $\cup$ operator horizontally concatenates the integer *root* to an existing sequence of integers.

For BUILD_S$_l$, we use the exponent $k_{1\alpha}$ on the first variable in each polynomial to construct a full set of possible sequences for that polynomial. Those sets of sequences are then combined in SUM_SEQUENCES (alg. 2) using a Cartesian product over the sets in each multinomial.

When calculating multinomial coefficients, we use the form in Eq. (13) in terms of binomial coefficients with a fast, stable algorithm from Manolopoulos [2002].

In practice, many of the group operations π produce identical products $M_1^{r_1} M_2^{r_2} \dots M_m^{r_m}$. Thus before computing any of the coefficients from the polynomials, we first form the polynomial products for each group operation and then add identical products together.

4. COMPUTATIONAL ORDER AND PERFORMANCE

The algorithm is structured around the *a priori* knowledge of the target stoichiometry. At the earliest possibility, we prune terms from individual polynomials that would not contribute to the final Pólya coefficient in the expanded product of polynomials (see Figure 3). Because the Pólya polynomial for each group operation is based on its disjoint-cyclic form, the complexity of the search can vary drastically from one group operation to the next. That said, it is common for groups to have several classes whose group operations (within each class) will have similar disjoint-cyclic forms and thus also scale similarly. However, from group to group, the set of classes and disjoint-cyclic forms may differ considerably; this makes it difficult to make a statement about the scaling of the algorithm in general. As such, we instead provide a formal, worst-case analysis for the algorithm's performance and supplement it with experimental examples. For these experiments, we crafted special groups with specific properties to demonstrate the various scaling behaviors as group properties change.

4.1. Worst-Case Scaling

Heuristically, the behavior of our algorithm should depend roughly on the size of the group: the number of permutations we have to analyze. That seems consistent with our experiments. But that can also be mitigated by noting that some groups of the same size have many more distinct cycle types than others. For example, if our group is generated by a single cycle of prime integer length p, then there are only two cycle types, despite the group having order p.

The majority of computation time should be spent in enumerating those matrices S and be proportional to the number of same (see Figure 4). Numerical experiments

ALGORITHM 1: Recursive Sequence Constructor

```
Procedure initialize(i, $k_{i\alpha}$, $M_\alpha^{r_\alpha}$, $V_\alpha$, T)
    Constructs a Sequence Object tree recursively for a single $M_\alpha^{r_\alpha}$ by filtering possible exponents
    on each $x_i$ in the polynomial. The object has the following properties:
      root: $k_{i\alpha}$, proposed exponent of $x_i$ in $M_\alpha^{r_\alpha}$.
      parent: proposed Sequence for $k_{i-1,\alpha}$ of $x_{i-1}$.
      used: the sum of the proposed exponents to left of and including this variable $\sum_{j=1}^{i} k_{i\alpha}$.
      $i$: index of variable in $M_\alpha^{r_\alpha}$ (column index).
      $k_{i\alpha}$: proposed exponent of $x_i$ in $M_\alpha^{r_\alpha}$ (matrix entry at $i\alpha$).
      $M_\alpha^{r_\alpha}$: Pólya polynomial in $P_\pi$ (14).
      $V_\alpha$: possible exponents for $M_\alpha^{r_\alpha}$ (10).
      $\mathbf{T}$: $\{T_i\}_{i=1}^{\xi}$ target stoichiometry.
    .....................................................................
    if $i = 1$ then
        self.used ← self.root + self.parent.used
    else
        self.used ← self.root
    end
    self.kids ← empty
    if $i \le \xi$ then
        for $p \in V_\alpha$ do
            $rem \leftarrow p - self.\text{root}$
            if $0 \le rem \le T_i$ and $|rem| \le d_\alpha r_\alpha - self.used$ and $|p - self.used| \bmod r_\alpha = 0$ then
                self.kids ⇐ initialize($i+1, rem, M_\alpha^{r_\alpha}, V_\alpha, \mathbf{T}$)
            end
        end
    end
Function expand(sequence)
    Generates a set of $S_{l\alpha}$ from a single Sequence object.
      sequence: the object created using initialize().
    .....................................................................
      sequences ← empty
    for kid ∈ sequence.kids do
        for seq ∈ expand(kid) do
            sequences ⇐ kid.root ∪ seq
        end
    end
    if len(sequence.kids) = 0 then
        sequences ← {kid.root}
    end
    return sequences
Function build_$S_l$($\mathbf{k}$, $\mathbf{V}$, $P_\pi$, $\mathbf{T}$)
    Constructs $S_l$ from $\{k_{1\alpha}\}_{\alpha=1}^{m}$ for a $P_\pi$ (14).
      $\mathbf{k}$: $\{k_{1\alpha}\}_{\alpha=1}^{m}$ set of possible exponent values on the first variable in each $M_\alpha^{r_\alpha} \in P_\pi$.
      $\mathbf{V}$: $\{V_\alpha\}_{\alpha=1}^{m}$ possible exponents for each $M_\alpha^{r_\alpha}$ (10).
      $P_\pi$: Pólya polynomial representation for a single operation $\pi$ in the group $\mathbf{G}$ (14).
      $\mathbf{T}$: $\{T_i\}_{i=1}^{\xi}$ target stoichiometry.
    .....................................................................
      sequences ← empty
    for $\alpha \in \{1 \dots m\}$ do
        seq ← initialize($1, k_{1\alpha}, M_\alpha^{r_\alpha}, V_\alpha, \mathbf{T}$)
        sequences ⇐ expand(seq)
    end
    return sequences
```

ALGORITHM 2: Coefficient Calculator

```
Function sum_sequences($S_l$)
    Finds $\tau_l$ (17) for $S_l = \{S_{l\alpha}\}_{\alpha=1}^{m}$ (16)
    S$_l$: a set of lists (of exponent sequences $\{k_{i\alpha}\}_{i=1}^{\xi}$) for each polynomial $M_\alpha^{r_\alpha}$ in $P_\pi$ (14).
    ..........................................................................
    $K_l \leftarrow S_{l1} \times S_{l2} \times \cdots \times S_{lm} = \langle\{(k_{i\alpha})_{i=1}^{\xi}\}_{\alpha=1}^{m}\rangle_l$
    coeff $\leftarrow 0$
    for each $\{(k_{i\alpha})_{i=1}^{\xi}\}_{\alpha=1}^{m} \in K_l$ do
        if $\sum_{\alpha=1}^{m} k_{i\alpha} = T_i \ \forall\, i \in \{1 \ldots \xi\}$ then
            coeff $\leftarrow$ coeff $+ \prod_{\alpha=1}^{m} \binom{d_\alpha}{\{k_{i\alpha}\}_{i=1}^{\xi}}$
        end
    end
    return coeff
Function coefficient(T, $P_\pi$, V)
    Constructs $\mathbf{S} = \{S_l\}$ and calculates $t_\pi$ (17)
    T: $\{T_i\}_{i=1}^{\xi}$ target stoichiometry.
    $P_\pi$: Pólya polynomial representation for a single operation $\pi$ in the group G (14).
    V: $\{V_\alpha\}_{\alpha=1}^{m}$ possible exponents for each $M_\alpha^{r_\alpha}$ (10).
    ..........................................................................
    if $m = 1$ then
        if $r_1 > T_i \ \forall\, i = 1..\xi$ then
            return 0
        else
            return $\binom{d_1}{T_1 T_2 \ldots T_\xi}$
        end
    else
        T $\leftarrow$ sorted(T)
        possible $\leftarrow V_1 \times V_2 \times \cdots \times V_m$
        coeffs $\leftarrow 0$
        for $\{k_{1\alpha}\}_{\alpha=1}^{m} \in$ possible do
            if $\sum_{\alpha=1}^{m} k_{1\alpha} = T_1$ then
                $S_l \leftarrow$ build_$S_l$($\{k_{1\alpha}\}_{\alpha=1}^{m}$, V, $P_\pi$, T)
                coeffs $\leftarrow$ coeffs+ sum_sequences($S_l$)
            end
        end
        return coeffs
    end
```

confirm[5] that the number of matrices scales exponentially with the number of colors (fixed group and number of elements in the set), linearly with the number of elements in the set (fixed number of colors and group), and is linear with the group size (fixed number of colors and elements in the set). The number of entries in the matrix S is $t\xi$ (see the discussion above Equation (3)) and the height of the entries is (roughly) bounded by the number of cycles and (very roughly) by the color frequencies divided by cycle lengths. This makes computing a time estimate based on these factors very difficult, but in the worst case, it could grow like the $t\xi$-th power of the average size of the entries, which will depend on the size of the target frequencies, and so on. This would be a very complex function to estimate, but we may expect it to grow exponentially for

[5]Figures are included in the code repository. See supplementary material.

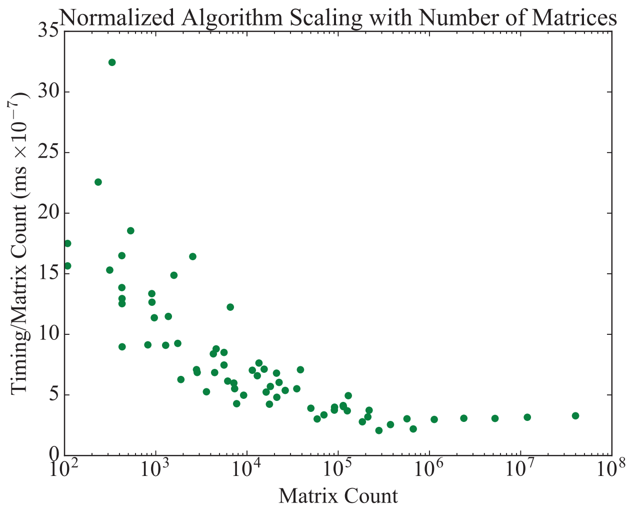


Fig. 4. Normalized algorithm scaling with the number of relevant matrices to enumerate. For large matrix counts, the behavior appears linear, supporting the hypothesis that the algorithm scales roughly with the number of matrices. The scatter is appreciable only for small matrix counts (less than 10^6).

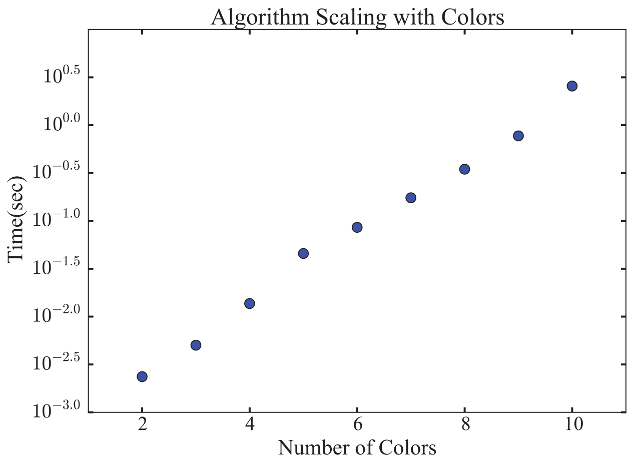


Fig. 5. Log plot of the algorithm scaling as the number of colors increases. Since the number of variables x_i in each polynomial increases with the number of colors, the combinatoric complexity of the expanded polynomial increases drastically with each additional color; this leads to an exponential scaling. The linear fit to the logarithmic data has a slope of 0.403.

very large input. We did not find that to be an impediment for the sizes of problems we needed to solve.

4.2. Experiments Demonstrating Algorithm Scaling

In Figure 5, we plot the algorithm's scaling as the number of colors in the enumeration increases (for a fixed group and number of elements). For each r-cycle in the disjoint-cyclic form of a group operation, we construct a polynomial with ξ variables, where ξ is the number of colors used in the enumeration. Because the group operation results in a product of these polynomials, increasing the number of colors by 1 increases the combinatoric complexity of the polynomial *expansion* exponentially. For

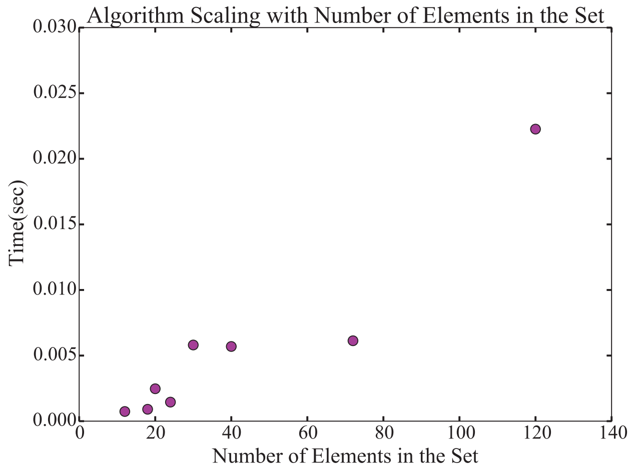


Fig. 6. Algorithm scaling as the number of elements in the finite set increases (for two colors). The Pólya polynomial arises from the group operations' disjoint-cyclic form, so more elements in the set results in a richer spectrum of possible polynomials multiplied together. Because of the algorithms aggressive pruning of terms, the exact disjoint-cyclic form of individual group operations has a large bearing on the algorithm's scaling. As such, it is not surprising that there is some scatter in the timings as the number of elements in the set increases.

this scaling experiment, we used the same transitive group acting on a finite set with 20 elements for each data point but increased the number of colors in the fixed color term T. We chose T by dividing the number of elements in the group as equally as possible; thus for two colors, we used $[10, 10]$; for three colors we used $[8, 6, 6]$, then $[5, 5, 5, 5]$, $[4, 4, 4, 4, 4]$, and so on. Figure 5 plots the $\log_{10}$ of the execution time (in ms) as the number of colors increases. As expected, the scaling is linear (on the log plot).

As the number of elements in the finite set increases, the possible Pólya polynomial representations for each group operation's disjoint-cyclic form increases exponentially. In the worst case, a group acting on a set with k elements may have an operation with k 1-cycles; on the other hand, that same group may have an operation with a single k-cycle, with lots of possibilities in between. Because of the richness of possibilities, it is almost impossible to make general statements about the algorithm's scaling without knowing the structure of the group and its classes. In Figure 6, we plot the scaling for a set of related groups (all are isomorphic to the direct product of $S_3 \times S_4$) applied to finite sets of varying sizes. Every data point was generated using a transitive group with 144 elements. Thus, this plot shows the algorithm's scaling when the group is the same and the number of elements in the finite set changes. Although the scaling appears almost linear, there is a lot of scatter in the data. Given the rich spectrum of possible Pólya polynomials that we can form as the set size increases, the scatter is not surprising.

Finally, we consider the scaling as the group size increases (Figure 7). For this test, we selected the set of unique groups arising from the enumeration of all derivative super structures of a simple cubic lattice for a given number of sites in the unit cell [Hart and Forcade 2008]. Since the groups are formed from the symmetries of real crystals, they arise from the semidirect product of operations related to physical rotations and translations of the crystal. In this respect, they have similar structure for comparison. In most cases, the scaling is obviously linear; however, the slope of each trend varies from group to group. This once again highlights the scaling's heavy dependence on

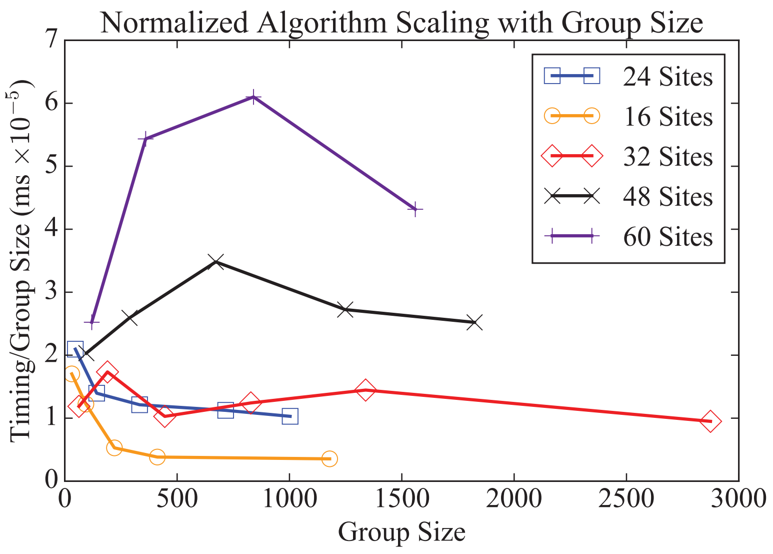


Fig. 7. Normalized algorithm scaling with group size for an enumeration problem from solid state physics [Hart and Forcade 2008]. We used the unique permutation groups arising from all derivative super structures of a simple cubic lattice for a given number of sites in the unit cell. The behavior is generally linear with increasing group size.

the specific disjoint-cyclic forms of the group operations. Even for groups with obvious similarity, the scaling may differ.

4.3. Comparison with Computer Algebra Systems

In addition to the explicit timing analysis and experiments presented above, we also ran a group of representative problems with our algorithm and MATHEMATICA (a common CAS). We also attempted the tests with MAPLE but were unable to obtain consistent results between multiple runs of the same problems.[6] So, we have opted to exclude the MAPLE timing results. For the comparison with MATHEMATICA, we used MATHEMATICA's `Expand` and `Coefficient` functions to return the relevant coefficient from the Pólya polynomial (see Figure 8).

5. SUMMARY

Until now, no low-level, numerical implementation of Pólya's enumeration theorem has been readily available; instead, a CAS was used to symbolically solve the polynomial expansion problem posed by Pólya. While CAS's are effective for smaller, simpler calculations, as the difficulty of the problem increases, they become impractical solutions. Additionally, codes that perform the actual enumeration of the colorings are often implemented in low-level codes, and interoperability with a CAS is not necessarily easy to automate.

We presented a low-level, purely numerical algorithm and code that exploits the properties of polynomials to restrict the combinatoric complexity of the expansion. By considering only those coefficients in the unexpanded polynomials that might contribute to the final answer, the algorithm reduces the number of terms that must be included to find the significant term in the expansion.

[6]The inconsistency manifests in MAPLE sometimes returning 0 instead of the correct result and sometimes running the same problem unpredictably in hours or seconds.

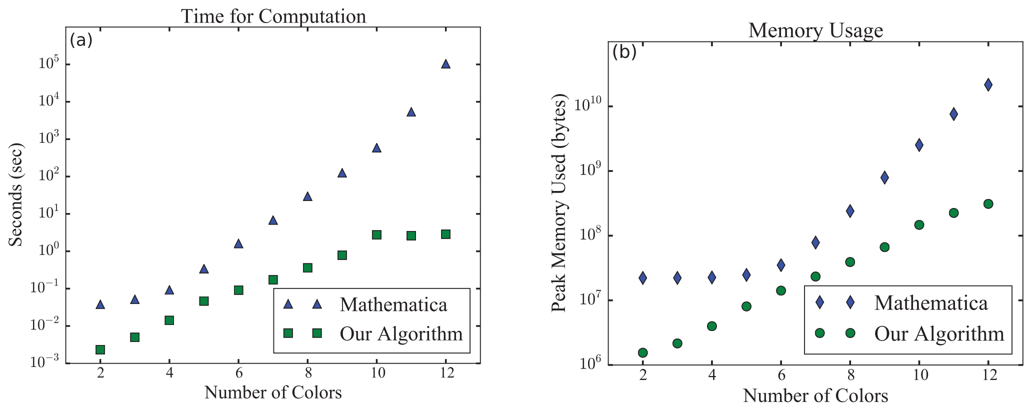


Fig. 8. Comparison of the CPU time (a) and memory usage (b) between the FORTRAN implementation of our algorithm and MATHEMATICA as the number of colors increases. These are the times needed to generate the data in Figure 5.

Because of the algorithm scaling's reliance on the exact structure of the group and the disjoint-cyclic form of its operations, a rigorous analysis of the scaling is not possible without knowledge of the group. Instead, we presented some numerical timing results from representative, real-life problems that show the general scaling behavior.

In contrast to the CAS solutions whose execution times range from milliseconds to hours, our algorithm consistently performs in the millisecond to second regime, even for complex problems. Additionally, it is already implemented in both high- and low-level languages, making it useful for confirming enumeration results. This makes it an effective substitute for alternative CAS implementations.

REFERENCES

Stefano Curtarolo, Gus L. W. Hart, Marco Buongiorno Nardelli, Natalio Mingo, Stefano Sanvito, and Ohad Levy. 2013. The high-throughput highway to computational materials design. *Nat. Mater.* 12, 3 (MAR 2013), 191–201. DOI:http://dx.doi.org/10.1038/NMAT3568

Kecai Deng and Jianguo Qian. 2014. Enumerating stereo-isomers of tree-like polyinositols. *J. Math. Chem.* 52, 6 (2014), 1581–1598.

Roberto Dovesi, Roberto Orlando, Alessandro Erba, Claudio M. Zicovich-Wilson, Bartolomeo Civalleri, Silvia Casassa, Lorenzo Maschio, Matteo Ferrabone, Marco De La Pierre, Philippe D'Arco, Yves Nol, Mauro Caus, Michel Rrat, and Bernard Kirtman. 2014. CRYSTAL14: A program for the ab initio investigation of crystalline solids. *Int. J. Quant. Chem.* 114, 19 (2014), 1287–1317. DOI:http://dx.doi.org/10.1002/qua.24658

Antoine Genitrini, Bernhard Gittenberger, Veronika Kraus, and Cécile Mailler. 2015. Associative and commutative tree representations for Boolean functions. *Theor. Comput. Sci.* 570 (2015), 70–101.

Modjtaba Ghorbani and Mahin Songhori. 2014. The enumeration of Chiral isomers of tetraammine platinum (II). *Match-Communications in Mathematical and in Computer Chemistry* 71, 2 (2014), 333–340.

Frank Harary. 1955. The number of linear, directed, rooted, and connected graphs. *Trans. Am. Math. Soc.* 78, 2 (1955), 445–463.

Gus L. W. Hart and Rodney W. Forcade. 2008. Algorithm for generating derivative structures. *Phys. Rev. B* 77 (Jun 2008), 224115. Issue 22. DOI:http://dx.doi.org/10.1103/PhysRevB.77.224115

Gus L. W. Hart and Rodney W. Forcade. 2009. Generating derivative structures from multilattices: Application to HCP alloys. *Phys. Rev. B* 80 (July 2009), 014120.

Gus L. W. Hart, Lance J. Nelson, and Rodney W. Forcade. 2012. Generating derivative structures for a fixed concentration. *Comp. Mat. Sci.* 59 (2012), 101–107. DOI:http://dx.doi.org/10.1016/j.commatsci.2012.02.015

B. A. Kennedy, D. A. McQuarrie, and C. H. Brubaker Jr. 1964. Group theory and isomerism. *Inorg. Chem.* 3, 2 (1964), 265–268.

Peter Lackner, Harald Fripertinger, and Gerhard Nierhaus. 2015. Peter Lackner/tropical investigations. In *Patterns of Intuition*. Springer, Berlin, 279–313.

Yannis Manolopoulos. 2002. Binomial coefficient computation: Recursion or iteration? *ACM SIGCSE Bulletin InRoads* 34 (Dec 2002). Issue 4. DOI:http://dx.doi.org/10.1145/820127.820168

James McGrane, Sanjaye Ramgoolam, and Brian Wecht. 2015. Chiral ring generating functions & branches of moduli space. *arXiv preprint arXiv:1507.08488* (2015).

Sami Mustapha, Philippe DArco, Marco De La Pierre, Yves Nol, Matteo Ferrabone, and Roberto Dovesi. 2013. On the use of symmetry in configurational analysis for the simulation of disordered solids. *J. Phys.: Condens. Matter* 25, 10 (2013), 105401. http://stacks.iop.org/0953-8984/25/i=10/a=105401.

George Pólya. 1937. Kombinatorische anzahlbestimmungen fr gruppen, graphen und chemische verbindungen. *Acta Math.* 68, 1 (1937), 145–254.

George Pólya and Ronald C. Read. 1987. *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds* (1987).

Jianguo Qian. 2014. Enumeration of unlabeled uniform hypergraphs. *Discr. Math.* 326, 1 (2014), 66–74.

R. W. Robinson, F. Harry, and A. T. Balaban. 1976. The numbers of chiral and achiral alkanes and monosubstituted alkanes. *Tetrahedron* 32, 3 (1976), 355–361.

Masahiko Taniguchi, Sarah Henry, Richard J. Cogdell, and Jonathan S. Lindsey. 2014. Statistical considerations on the formation of circular photosynthetic light-harvesting complexes from rhodopseudomonas palustris. *Photosynth. Res.* 121, 1 (2014), 49–60.

J. Tura, R. Augusiak, A. B. Sainz, B. Lücke, C. Klempt, M. Lewenstein, and A. Acín. 2015. Nonlocality in many-body quantum systems detected with two-body correlators. *arXiv preprint arXiv:1505.06740* (2015).

Received December 2015; revised May 2016; accepted June 2016

CHAPTER 3

Enumerating Derivative Superstructures

Derivative superstructures [71] are structures whose lattice vectors are multiples of a "parent lattice" and whose atomic basis vectors are constructed from the the lattice points of the parent lattice. As mentioned in Chapter 2 numerous algorithms exist for enumerating unique arrangements of atoms within a derivative superstructure. However, these algorithms often run out of memory when the number of possible arrangements of atoms is large, such as with high-entropy alloys (HEAs).

The following paper contains an algorithm that overcomes this problem by:

1. Using the Pólya enumeration algorithm from Chapter 2 to determine the number of unique arrangements before enumerating and then checks if they will fit into memory.

2. Using a tree search algorithm to only construct the list of unique arrangements without constructing any duplicates (most algorithms construct a complete list of all the possible arrangements then "cross out" the duplicates).

This offers advantages over other implemented algorithms as it avoids the combinatoric explosion of possible arrangements that is often much larger than the number of unique arrangements. For example, the HEA considered in Section 1.2 that consists of 5 atomic species with equal concentration within a 20 atom cell of an fcc lattice has 3×10^{12} possible atomic arrangements, however, it has only 10^9 unique arrangements. For this system the new algorithm will only require the memory needed to store the unique list of 10^9 arrangements while other algorithms will need three orders of magnitude more memory to store the list of possible arrangements.

This efficient enumeration also allows for displacement directions to be included in the enumeration. The addition of displacement directions increases the number of possible arrangements of atoms on the lattice by $6n!$ where n is the number of atoms that can be displaced off the lattice. A complete list of possible arrangements, including displacement directions, would exceed computer memory for all but the smallest of systems. This makes the new enumeration algorithm the only algorithm that can handle the additional complexity of displacement directions.

Being able to include displacement directions in the enumerated list of unique arrangements allows for the enumeration of systems that can be used to study site-disordered solids [72] or any system for which atoms are displaced from ideal lattice sites. This algorithm has been implemented in the `enumlib` package and in the `Python` package "phenum" available on pypi or as source code at https://github.com/wsmorgan/phonon-enumeration/.

For this article, I wrote the algorithm in Python and FORTRAN and implemented tests for the algorithm in both languages and wrote the bulk of the text for the article. Dr. Rodney Forcade designed the algorithm with input from me and my advisor. All the authors contributed to the text.

The following article is reproduced with permission. A license can be found in Appendix E.

Computational Materials Science 136 (2017) 144–149

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Generating derivative superstructures for systems with high configurational freedom

Wiley S. Morgan [a],[*], Gus L.W. Hart [a], Rodney W. Forcade [b]

[a] Department of Physics and Astronomy, Brigham Young University, Provo, UT 84602, USA
[b] Department of Mathematics, Brigham Young University, Provo, UT 84602, USA

ARTICLE INFO

Article history:
Received 5 January 2017
Received in revised form 11 April 2017
Accepted 16 April 2017
Available online 29 May 2017

Keywords:
Enumeration
Derivative superstructures
Displacement patterns

ABSTRACT

Modeling alloys requires the exploration of all possible configurations of atoms. Additionally, modeling the thermal properties of materials requires knowledge of the possible ways of displacing the atoms. One solution to finding all symmetrically unique configurations and displacements is to generate the complete list of possible configurations and remove those that are symmetrically equivalent. This approach, however, suffers from a combinatorial explosion when the supercell size is large, when there are more than two atom types, or when there are many displaced atoms. This problem persists even when there are only a relatively small number of unique arrangements that survive the elimination process. Here, we extend an existing algorithm to include the extra configurational degrees of freedom from the inclusion of displacement directions. The algorithm uses group theory and a tree-like data structure to eliminate large classes of configurations, avoiding the typical combinatoric explosion. With this approach we can now enumerate previously inaccessible cases, including atomic displacements.

1. Introduction

In computational material science, one frequently needs to list the “derivative superstructures” [1] of a given lattice. A derivative superstructure is a structure with lattice vectors that are multiples of a “parent lattice” and have atomic basis vectors constructed from the lattice points of the parent lattice. For example, many phases in metal alloys are merely “superstructures” of fcc, bcc, or hcp lattices ($L1_0$, $L1_2$, B2, $D0_{19}$, etc.). When modeling alloys, it is necessary to explore all possible configurations and concentrations of atoms within these superstructures. When determining if a material is thermodynamically stable, the energies of the unique arrangements are compared to determine which has the lowest energy.

Derivative superstructures are found using combinatoric searches [2–8], comparing every possible combination of atoms to determine which are unique. However, these searches can be computationally expensive for systems with high configurational freedom and are sometimes impractical due to the combinatoric explosion of possible arrangements.

Other problems impaired by the inefficiency of current enumeration methods include modeling materials that have disorder in their structures, such as site-disordered solids [9] or that include atomic displacements as a degree of freedom [10–12]. There are numerous techniques available for modeling these systems including cluster expansion (CE) [13] and a recently developed “small set of ordered structures” (SSOS) method [14]. However, the accuracy of these methods is still linked to the number of unique configurations being modeled. In other words, if the model is trained on a small set of configurations then it will not be able to make accurate predictions. Increasing the number of configurations used to train the models can improve their predictive power. Increasing the number of structures being used requires a more efficient enumeration technique than those currently available.

Leveraging the basic concepts of the algorithm presented in Ref. [6], we altered the algorithm to have more favorable scaling in multinary cases. The basic idea is to imagine the enumeration as a tree search and employ two new ideas: (1) “partial colorings” and (2) stabilizer subgroups. Section 3 illustrates the algorithm with a concrete example.

The concept of partial colorings is to skip entire branches of the tree that are symmetrically equivalent to previously visited branches. A partial coloring is an intermediate level in the tree (see Fig. 1) where configurations are not yet completely specified. It frequently happens that symmetric redundancy can be identified at an early, “partially colored” stage, avoiding the need to descend further down the tree.

* Corresponding author.
E-mail address: wiley.s.morgan@gmail.com (W.S. Morgan).

http://dx.doi.org/10.1016/j.commatsci.2017.04.015
0927-0256/© 2017 Elsevier B.V. All rights reserved.

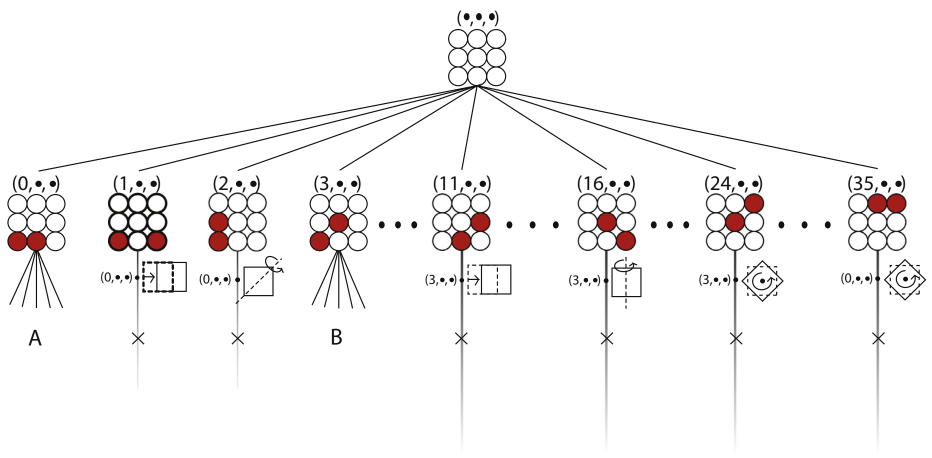


Fig. 1. The empty lattice and 8 of the 36 configurations with only red atoms are shown for the example discussed in Section 3. Above each partial coloring is a vector that indicates its location in the tree, i.e. (x_r, x_y, x_p), where the x_is are integers that indicate which arrangement of that color is on the lattice and a • means that no atoms of that color have been placed yet. Below each configuration is either the label of a symmetrically equivalent configuration, along with the group operation that makes them equivalent, or the letters A and B. A and B are the branches that are built from the 1-partial colorings that are unique and are displayed in Fig. 2. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Stabilizer subgroups [15] further increase the efficiency of the new algorithm. The stabilizer subgroup at each stage is the set of symmetries which leave the current partial coloring unchanged. As we add more colors, and eliminate symmetrically equivalent colorings, we need not consider colorings which would be equivalent by non-stabilizer symmetries, since those colorings have already been implicitly eliminated. Note that the stabilizer subgroup will get smaller as we proceed down the search tree, thus simplifying and speeding our search.

2. Supercell selection and the symmetry group

The first step in enumerating derivative superstructures is the enumeration of unique supercells. This step was solved in Ref. [8], but due to its importance to the algorithm we provide a brief overview.

The supercells, of size n, are found by constructing all Hermite Normal Form (HNF) matrices whose determinant is n. An HNF matrix is an integer matrix with the following form and relations:

$$\begin{pmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{pmatrix}, \; 0 \leqslant b < c, \; 0 \leqslant d < f, \; e < f \tag{1}$$

where $acf = n$. The HNFs determine all possible the supercells for the system. For example, consider a 9-atom cell, then $n = 9$ and a, c, f are limited to permutations of $(1,3,3)$ and $(1,1,9)$. Then following the rules for the values of b, d, and e, every HNF for this system can be constructed. These HNFs represent all the possible supercells of size n of the selected lattice. Some of these are equivalent by symmetry, so the symmetry group of the parent lattice is used to eliminate any duplicates.

Next, we convert the symmetries of the lattice to a list of permutations of atomic sites. There is a one-to-one mapping between the symmetries of the lattice and atomic site permutations, i.e., the groups are isomorphic. The mapping from the symmetry operations to the permutation group is accomplished using the quotient group $G = L/L'$, where L is the lattice, constructed from the unit cell, and L' is the superlattice, constructed from the supercell. The quotient group G is found directly from the Smith Normal Form (SNF) matrices, which can be constructed from the HNFs via a standard algorithm using integer row and column operations. Thus $S = UHV$ where U and V are integer matrices with determinant ± 1 and S is the diagonal SNF matrix, where each positive integer diagonal entry divides the next one down. The group, G, is then $G = Z_{s_1} \oplus Z_{s_2} \oplus Z_{s_3}$, where s_i is ith diagonal of the SNF and Z_{s_i} represents the cyclic group of order n.

Once the supercells have been found and their symmetry groups have been converted to the isomorphic permutation group, the algorithm can begin finding the unique arrangements of atoms within each supercell in a tree search framework. This is accomplished by treating each supercell with its symmetry group as a separate enumeration problem. The results of the enumeration across all supercells are then combined to produce the full enumeration.

3. Tree search

Once a supercell has been selected, the remainder of the enumeration algorithm resembles a tree search. It is often possible to skip the descendents of a node because we know all its "leaves" will represent duplicate structures. These nodes represent incomplete configurations, or *partial colorings* (see Figs. 1 and 2). The partial colorings are identified using a "location vector" — a list of indices that specify the node in the tree. Once a partial coloring is constructed, the stabilizer subgroup for that partial coloring is found. The stabilizer subgroup allows for the comparison of branches within the tree in a manner that minimizes the number of group operations used. These tools (partial colorings and the stabilizer subgroup) are used to "prune" branches of the tree as they are being constructed, eliminating large classes of arrangements at once (Fig. 3).

We will use a 2D example of a 9-atom cell to illustrate the algorithm. The lattice will be populated with the following atomic species; 2 red atoms, 3 yellow atoms, and 4 purple atoms. A subset of the possible arrangements of this system is shown in Fig. 2. The concepts illustrated with this 2D example are equally applicable in 3D.

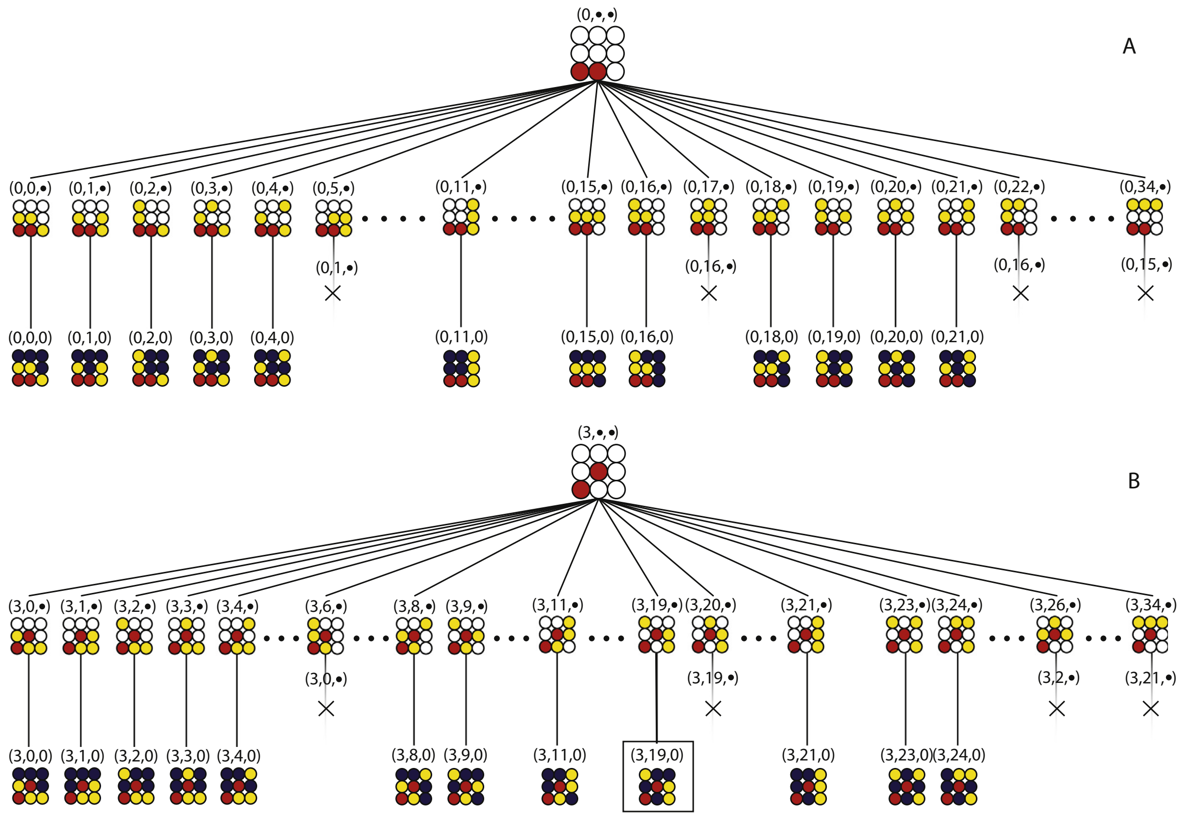


Fig. 2. Here the A and B branches of the tree from Fig. 1 are shown. Each branch starts with the initial 1-partial coloring the branch is built from ($(0,\bullet,\bullet)$ and $(3,\bullet,\bullet)$ respectively). The branches then show a selection of the 2-partial colorings for that branch, and the unique full colorings that are found. As in Fig. 1 the vectors that indicate the configurations location in the tree are displayed above the configurations and the symmetrically equivalent labels appears beneath them. In this figure the actions that make the configurations have been excluded due to their complexity. For example, The configuration labeled $(0,5,\bullet)$ is equivalent to the $(0,1,\bullet)$ configuration by a rotation about the vertical followed by a translation to the left. In the B branch configuration $(3,19,0)$ is outlined for reference because it is used as an example later in the text. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

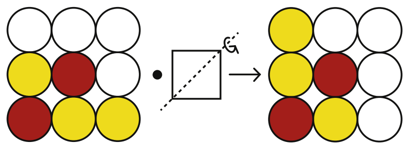


Fig. 3. The configuration $(3,0,\bullet)$, shown on the left, is acted on by a reflection about the diagonal resulting in configuration $(3,6,\bullet)$, shown on the right. Because the symmetry group operation is a stabilizer for the configuration $(3,\bullet,\bullet)$ the red atoms were not affected. A stabilizer is a group element that leaves the set invariant. The yellow atoms, however, were mapped to a different configuration. This means we can use just the stabilizer subgroup for the $(3,\bullet,\bullet)$ configuration to compare all the 2-partial colorings of the form $(3,x_y,\bullet)$, where $(0 \leqslant x_y \leqslant C_y - 1)$, because any other group operation would map us to a different branch of the tree. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.1. Partial colorings

When searching for all unique configurations, it is useful to know, a priori, how many configurations are expected. A recently developed numerical algorithm for the Pólya enumeration theorem [16–18] allows one to quickly determine the memory requirements of storing the unique arrangements. For the 9 atom system considered here, the Pólya algorithm finds 24 unique configurations (from the 1260 combinatorially distinct configurations).

The algorithm places atomic species on the lattice from smallest concentration to the largest. In this case, the red atoms have the lowest concentration and are placed in the first two sites of the cell creating the first 1-partial coloring (a partial coloring is a configuration with only a subset of the atoms decorating the lattice). This is shown in the leftmost configuration, labeled $(0,\bullet,\bullet)$, in the second row of Fig. 1. The general procedure is to apply the symmetry group to each partial coloring in order to make comparisons between partial colorings and determine if they are symmetrically equivalent. For example, in Fig. 1, the configuration labeled $(1,\bullet,\bullet)$ is equivalent to configuration $(0,\bullet,\bullet)$ by a translation. At the $(0,\bullet,\bullet)$ stage we only have one partial coloring so it is unique and no comparisons need to be made, however the symmetry group is still applied to find the stabilizer subgroup described in Section 3.2.

Comparisons between configurations are made by using a hash function. The hash function is a one-to-one mapping between the atomic configurations and the location vector. In our case, the configurations are listed within the hash table in the order they are created. The hash function then maps the configuration to a vector of integers with an entry for each species (color) in the system.

The hash function uses the principles of combinatorics to uniquely identify each partial coloring using an integer vector. Its construction starts by determining the number of possible ways to arrange the colors on the lattice. The number of possible configurations can be found using the multinomial coefficient, which is equivalent to the product of binomial coefficients for each individual color:

$$C = \binom{n}{a_1, a_2, \ldots, a_k} = C_1 C_2 \ldots C_k = \binom{n}{a_1}\binom{n-a_1}{a_2} \ldots \binom{n - a_1 - a_2 - \ldots - a_k}{a_k}, \tag{2}$$

where n is the number of sites in the unit cell and $a_1, a_2, \ldots, a_k$ are the number of atoms of species i such that $\sum_i a_i = n$. The binomials determine the number of ways to place the atoms of each color within the lattice once the previous colors have been placed. By assigning each partial coloring an integer, x_i, from 0 to $C_i - 1$, where i is the color, we can build a vector that identifies the location, $(x_1, x_2, \ldots, x_k)$, of the configuration within the tree. For example, there are $C_r = \binom{9}{2} = 36$ ways to place the red atoms (r) on the empty lattice. After the red atoms are placed then there remain $C_y = \binom{7}{3} = 35$ ways to place the yellow atoms (y) on the remaining lattice sites. This leaves $C_p = \binom{4}{4} = 1$ way to place the purple atoms (p) on the lattice. Within Figs. 1 and 2, the vector locations have the form (x_r, x_y, x_p) and if the color has not been assigned yet then the as-yet-unspecified x_is are indicated by dots.

The hash function maps a configuration to a location vector as follows: the x_i's are constructed by considering each color separately and building a binary string of the color and the remaining empty lattice sites, where the color is a 1 and the empty site is a 0 within the string. From the binary string, we can then use a series of binomial coefficients to find the x_i's. The binomial coefficients are found by taking each 0 in the string that has 1's to the right of it and computing $\binom{p}{q-1}$, where p is the number of digits to the right and q is the number of 1's to the right of the 0. Summing the binomials for qualifying zeros produces a number that tells us how many configurations came before the current one.

As an example, consider configuration (3,19,0) of Fig. 2B. The construction begins with the red atoms represented as the following binary string (1,0,0,0,1,0,0,0,0), where every atom that is not red has been represented by a 0 and the red atoms by a 1. This string has 3 zeros that have a single 1 to their right, the first zero has 7 digits to its right, the second has 6 atoms to its right and the third has 5 atoms to its right. The resultant sum of binomials is $x_r = \binom{7}{0} + \binom{6}{0} + \binom{5}{0} = 1 + 1 + 1 = 3$. This result is the first entry in our location vector.

The second entry in the location vector is constructed for the yellow atoms. The bit string representation of the yellow atoms is (0,1,0,1,1,0,0), there are only 7 digits because the 2 red atoms have already been placed, so $x_y = \binom{6}{2} + \binom{4}{1} = 15 + 4 = 19$. The last entry in the location vector is built for the purple atoms which have the bit string (1,1,1,1), so $x_p = 0$. The location vector is complete once all colors have been included.

The location vectors allow us to determine if a configuration is unique by checking if an element of the symmetry group maps the configuration to another configuration with a *smaller* location vector. A symmetry operation maps a configuration's location to a second, equivalent location. Uniqueness is determined by comparing the original and mapped locations for the configuration; if the mapped configuration has already been enumerated, that is, if $x_{\text{original}} > x_{\text{mapped}}$, then the configuration is not unique because it is equivalent to one we have already visited. For example, configuration $(2, \bullet, \bullet)$ shown in Fig. 1 can be turned into configuration $(0, \bullet, \bullet)$ by a 180 degree rotation about the diagonal. Since $(2, \bullet, \bullet)$ and $(0, \bullet, \bullet)$ are equivalent we conclude that $(2, \bullet, \bullet)$ is not unique because $2 > 0$. In summary, if any element of the symmetry group makes the location vector "smaller", then the corresponding configuration has already been visited.

3.2. The stabilizer subgroup

The entire symmetry group does not need to be applied to a partial coloring; all that is needed is the stabilizer subgroup of the parent partial coloring (one level up the tree). The stabilizer subgroup is found when the symmetry group is applied to the partial coloring one level higher up the tree, so finding the stabilizer subgroup costs nothing computationally. As an example of an element of the stabilizer subgroup, consider the cell $(3, \bullet, \bullet)$, displayed in Fig. 1, and reflect it about the diagonal; the red atoms are unaffected. This means that a reflection about the diagonal is a member of the stabilizer subgroup for the 1-partial coloring $(3, \bullet, \bullet)$. In general, only a small subset of the symmetry group will be in the stabilizer subgroup for any partial coloring.

The stabilizer subgroup leaves the desired n-partial coloring unchanged, where n is the depth in the tree. When another color is added (making an $(n+1)$-partial coloring), the stabilizer subgroup for the n-partial coloring becomes the only group operations that can be applied without affecting the n-partial coloring. In other words, if we were to use any other group elements we would be comparing configurations that we already know are equivalent on the n-partial coloring level.

Once a unique n-partial coloring and its stabilizer subgroup have been found, the algorithm proceeds down the branch to the $(n+1)$-partial colorings (see Fig. 2). To check the uniqueness of the $(n+1)$-partial colorings, the stabilizer subgroup from the n-partial colorings are used. (At this point, the stabilizer subgroup for the $(n+1)$-partial colorings are stored.) When a unique configuration is found on the $(n+1)$ level, another color is added, making the $(n+2)$-partial colorings, and the process is repeated until the final level of the tree is reached.

The algorithm proceeds down a branch of the tree until a unique full configuration is found, such as (0,0,0) of Fig. 2. When the full configuration is found, the algorithm backs up one level and considers the next partial coloring. When no partial colorings are available on a level, the algorithm backs up until it finds a level with untested partial colorings. In this manner, the entire tree is explored but only sections with unique configurations are explored in detail.

For an example of the complete algorithm, consider Figs. 1 and 2. The algorithm starts at $(\bullet, \bullet, \bullet)$ then builds the 1-partial coloring at $(0, \bullet, \bullet)$, which is unique by virtue of being the first partial coloring considered on this level, and records its stabilizer subgroup. The yellow atoms are then added to the configuration to build the 2-partial coloring at $(0, 0, \bullet)$, of Fig. 2 A, which is also unique, and records its stabilizer subgroup. Next, it places the purple atoms to get the configuration at (0,0,0); this configuration is saved, then the algorithm backs up to the 2-partial coloring level to consider the configuration $(0, 1, \bullet)$ and find its stabilizer subgroup.

Once this process has been repeated for all 34 partial colorings in the vector $(0, x_y, \bullet)$ $(0 \leqslant x_y \leqslant 34 = C_y)$, the algorithm retreats to the 1-partial coloring level shown in the second row of Fig. 1 and finds that $(1, \bullet, \bullet)$ and $(2, \bullet, \bullet)$ are equivalent to $(0, \bullet, \bullet)$. It then begins to build the $(3, \bullet, \bullet)$ branch (Fig. 2 B) in the same manner as the $(0, \bullet, \bullet)$ branch. In this example, only 106 nodes of the 1296 are visited.

Since there are only two unique 1-partial colorings for this system the algorithm is complete once both branches that originate from these 1-partial colorings have been explored. In the end, 24 unique configurations are found (shown in Fig. 2A and

B), in agreement with the number determined by the Pòlya enumeration algorithm.

3.3. Extension to displacement degrees of freedom

Having established the algorithm, we will now address its extension to include displacement directions. These enumerations are more difficult because including displacement directions changes the action of the group. Displacement directions simply indicate the direction that an atom could be displaced off the lattice. The enumeration of structures that include displacement directions can be used to build databases [19] of possible structures with displacements included.

Our algorithm changes only slightly if displacement directions are included in the enumeration. First, the atoms that will be displaced are treated as a different atomic species so that each displaced atom's unique locations can be determined. (See Fig. 4 for an example where yellow displaced atoms are replaced with the red atoms from the example system used above.) Once the arrows have been replaced by atomic species, the algorithm proceeds as normal until a full configuration is found. The algorithm then restores the arrows and uses the stabilizer subgroup of the full configuration to check for equivalent arrow configurations.

In order to determine if the combined arrow and color configuration is unique, each group element has to be paired with a second set of permutations that determine how the symmetry operation affects the arrows. The effect on the arrows is represented as a permutation of the numbers 0 to $d-1$, where each number represents a different displacement direction up to the d directions being considered. For example, if we consider the system in Fig. 4, we have two atoms being displaced along one of the 6 cardinal directions, then any arrow could have values of between 0 and 5 where each integer has an associated direction; up = 0, right = 1, down = 2, left = 3, into the page = 4, and out of the page = 5. The initial arrow vector, shown in the figure, is (up, up) and is represented as $(0,0)$.

The comparison of the rotated and unrotated arrows is achieved using a hash function different from the one used to hash the color configurations. This hash function takes a vector of arrow directions $(a_0, a_1, a_2, \ldots, a_k)$ and converts it to a unique integer label (where a_i is an integer from 0 to $d-1$ indicating the direction of the ith arrow and $k+1$ is the number of arrows). The integer label is simply the mixed radix number:

$$x = \sum_{i=0}^{k} a_i d^i. \tag{3}$$

With the unique integer labels for each arrow arrangement, we can make comparisons between symmetry operations. As was the case for the configurations, if the effect of a symmetry operation results in a relationship of $x_{\text{old}} > x_{\text{new}}$, then the arrow configuration is not unique and can be skipped.

The stabilizer subgroup for the unique color configuration are used to map the arrows to new directions and the hash function is used to compare the original and mapped arrows. After an arrow arrangement is checked, the algorithm then increases the magnitude of the last a_k in the vector by 1 and checks it for uniqueness with the stabilizer subgroup. If increasing the magnitude of a_k would cause it to be greater than the value of $d-1$ then a_k becomes 0 again and a_{k-1} is increased by 1. This process is repeated until all the entries in the arrow vector are equal to $d-1$.

For example, the initial arrow vector for the system shown in Fig. 4 is (up,up) and is represented as $(0,0)$. It is found to be unique since it is the first arrangement. For the next arrangement the arrow on the right is rotated to point to the right creating the arrangement represented as $(0,1)$. This arrangement is also checked to see if it is unique. The right most arrow continues to be rotated every time a new arrangement is constructed until it is pointing out of the page and the arrangement represented as $(0,5)$ has been considered. At this point all possible arrangements that have the first arrow pointing up have been considered, so the second arrow is set to point up and first arrow is rotated to make the arrangement $(1,0)$. We then go back to increasing the last entry in the vector to create new arrangements in order to determine if any of them are unique until $(1,5)$ is reached. The process is repeated until all possible the arrangements, i.e., all 2-tuples of $0, 1, \ldots, d-1$, have been considered. Once all the vectors have been considered, the algorithm goes back up the tree to find the next unique configuration of colors.

In this manner, discrete displacement directions can be added to the configurations. In this example, adding two arrows to the system increases the number of possible arrangements to 45,360 (the number of possible arrangements for the atoms alone is 1260). However, the resultant number of unique arrangements is only 663.

4. Algorithm scaling

Our new algorithm is much more efficient because it does not compare all possible configurations of atoms to determine which are unique. To demonstrate this improvement, we explored a typical use case for the algorithm by finding all unique configurations of atoms on an fcc lattice. We considered ternary and quaternary

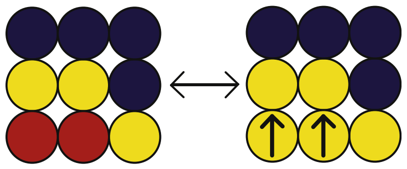


Fig. 4. To include displacement directions to the algorithm we represent the atoms to be displaced by a unique color and then convert them back once a unique configuration is found. In this figure two displaced yellow atoms are represented by red atoms until the previous portion of the algorithm is complete, then they are replaced by arrows again for the arrow enumeration. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

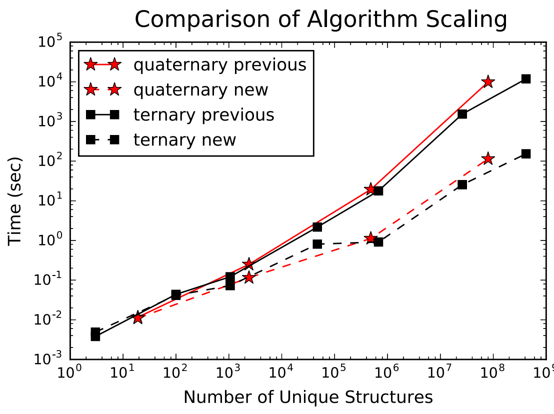


Fig. 5. The scaling of the new and previous algorithm for fcc ternary and quaternary systems in which the atomic species were of equal concentration. The ternary system was enumerated for cell sizes of 3–21 atoms and the quaternary system for cell sizes of 4–16 atoms.

systems in which all atoms were of equal concentrations. For the ternary system we explored cell sizes from 3 to 21 and for the quaternary system we considered cell sizes from 4 to 16. As can be seen in Fig. 5, the new algorithm is significantly faster (it scales better); for systems with approximately 20 atoms, the new algorithm is two orders of magnitude faster.

5. Conclusion

Our previous algorithms [6–8] explored configuration space by comparing all possible configurations of the atoms to eliminate those that were symmetrically equivalent. Our new algorithm uses a tree search and skips entire branches in the tree by employing partial colorings. Additionally, the stabilizer subgroups of the partial colorings increases the efficiency of comparing different configurations to determine which are unique. The new algorithm can be applied to cases where the previous algorithms failed because it handles cases with high configurational freedom more efficiently. This allowed us to extend it to include displacement directions.

With this new algorithm, it is now possible to find the unique arrangements of systems with higher configurational freedom. The systems now accessible include k-nary alloys, $k \geqslant 3$, and structures with displacement directions. This is accomplished by using an approach which closely resembles a tree search, but in which large classes of configurations are eliminated all at once. In this manner, we are able to partially avoid the combinatoric explosion which impedes the performance of the previous algorithms.

This algorithm has been implemented in the enumlib package and is available for public use at https://github.com/msg-byu/enumlib under the MIT License.

Acknowledgements

This research was funded by ONR grant MURI N00014-13-1-0635.

References

[1] M.J. Buerger, Derivative crystal structures, J. Chem. Phys. 15 (1) (1947) 1.
[2] P. D'Arco, S. Mustapha, M. Ferrabone, Y. Noël, M. De La Pierre, R. Dovesi, Symmetry and random sampling of symmetry independent configurations for the simulation of disordered solids, J. Phys. Condens. Matt. 25 (35) (2013) 355401.
[3] A. Van De Walle, G. Ceder, Automating first-principles phase diagram calculations, J. Phase Equilib. 23 (4) (2002) 348.
[4] A. Van De Walle, M. Asta, G. Ceder, The alloy theoretic automated toolkit: a user guide, Calphad 26 (4) (2002) 539–553.
[5] N.A. Zarkevich, T.L. Tan, D.D. Johnson, First-principles prediction of phase-segregating alloy phase diagrams and a rapid design estimate of their transition temperatures, Phys. Rev. B 75 (10) (2007) 104203.
[6] G.L.W. Hart, L.J. Nelson, R.W. Forcade, Generating derivative structures at a fixed concentration, Comput. Mater Sci. 59 (2012) 101–107.
[7] G.L.W. Hart, R.W. Forcade, Generating derivative structures from multilattices: algorithm and application to hcp alloys, Phys. Rev. B 80 (2009) 014120.
[8] G.L.W. Hart, R.W. Forcade, Algorithm for generating derivative structures, Phys. Rev. B 77 (2008) 224115.
[9] R. Grau-Crespo, S. Hamad, The symmetry-adapted configurational ensemble approach to the computer simulation of site-disordered solids, in: MOL2NET, International Conference on Multidisciplinary Sciences, page c002, Basel, Switzerland, December 2015. MDPI.
[10] S. Kadkhodaei, Q. Hong, A. van de Walle, Free energy calculation of mechanically unstable but dynamically stabilized bcc titanium, Phys. Rev. B 95 (2017) 064101.
[11] Z. Fei, W. Nielson, Y. Xia, V. Ozolins, Lattice anharmonicity and thermal conductivity from compressive sensing of first-principles calculations, Phys. Rev. Lett. 113 (18) (2014) 185501.
[12] K. Parlinski, Z.Q. Li, Y. Kawazoe, First-principles determination of the soft mode in cubic ZrO_2, Phys. Rev. Lett. 78 (1997) 4063–4066.
[13] J.M. Sanchez, F. Ducastelle, D. Gratias, Generalized cluster description of multicomponent systems, Phys. A 128 (1–2) (1984) 334–350.
[14] C. Jiang, B.P. Uberuaga, Efficient ab initio modeling of random multicomponent alloys, Phys. Rev. Lett. 116 (10) (2016) 105501.
[15] G.E. Edward, Transformation groups and C^*-algebras, Ann. Math. 81 (1) (1965) 38–55.
[16] G. Pólya, R.C. Read, Combinatorial enumeration of groups, graphs, and chemical compounds, Springer Science & Business Media, 2012.
[17] G. Pólya, Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen, Acta Math. 68 (1) (1937) 145–254.
[18] C.W. Rosenbrock, W.S. Morgan, G.L.W. Hart, S. Curtarolo, R.W. Forcade, Numerical algorithm for pólya enumeration theorem, J. Exp. Algorithm. 21 (1) (2016) 1–11.
[19] A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K.A. Persson, The Materials Project: a materials genome approach to accelerating materials innovation, APL Mater. 1 (1) (2013) 011002.

CHAPTER 4

Determining the Best Integration Grids

As mentioned in Section 1.3, a significant source of error in DFT calculations is from the **k**-point sampling for the electronic energy calculations. Errors in these calculations can affect every material property calculated using DFT and can be serious. Despite this, there have been no comprehensive studies on the behavior of the errors caused by **k**-points. Initial studies on semiconductors, such as Si, showed that increasing the number of **k**-points rapidly increases the accuracy of the calculation. The integral is less tame, however, for metals which have a discontinuity in the Fermi surface, see Fig 4.1. The discontinuities in the Fermi surface cause unpredictable jumps in the convergence of the energy calculations that can, at times, make choosing a denser sampling grid generate a less accurate answer. This means that to ensure accuracy **k**-point grids may need to be far denser than most researchers use.

DFT codes generally use *regular grids*, proposed by Monkhorst and Pack (MP) [3]. **k**-points of a regular grid are defined by:

$$\begin{aligned}\mathbf{k} &= (\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3)\mathbb{D}^{-1}\begin{pmatrix}n_1\\n_2\\n_3\end{pmatrix}\\ &= \frac{n_1}{d_1}\mathbf{b}_1+\frac{n_2}{d_2}\mathbf{b}_2+\frac{n_3}{d_3}\mathbf{b}_3\end{aligned} \tag{4.1}$$

where $\mathbf{b}_i$ are the reciprocal lattice vectors, $\mathbb{D}$ is any diagonal integer matrix whose diagonal elements are d_i, and n_i runs from 1 to d_i. These grids have become widely used because they are easy to implement.

An alternative, more general method was proposed by Moreno and Soler, [73] which involves searching through grids at a desired **k**-point density for those that have the fewest symmetrically distinct **k**-points. The grids are then sorted by the length of the shortest grid generating vector and the grid with the longest vector is chosen, thus selecting the most uniform grid. The Moreno-Soler method involves the construction of superlattices from the real-space parent lattice (primitive lattice)

$$(\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3) = (\mathbf{a}_1,\mathbf{a}_2,\mathbf{a}_3)\mathbb{N} \tag{4.2}$$

where $\mathbf{s}_i$ are the supercell vectors, $\mathbf{a}_i$ are the parent lattice vectors, and $\mathbb{N}$ is an integer matrix. The *dual* lattice of the supercell lattice then defines a set of **k**-point grid generating vectors $\boldsymbol{\kappa}_i$.

$$\begin{aligned}(\boldsymbol{\kappa}_1,\boldsymbol{\kappa}_2,\boldsymbol{\kappa}_3) &= 2\pi((\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3)^{-1})^T\\ &= 2\pi(((\mathbf{a}_1,\mathbf{a}_2,\mathbf{a}_3)\mathbb{N})^{-1})^T\\ &= 2\pi(\mathbb{N}^{-1})^T((\mathbf{a}_1,\mathbf{a}_2,\mathbf{a}_3)^{-1})^T\\ &= (\mathbb{N}^{-1})^T(\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3)\end{aligned} \tag{4.3}$$

Note that the determinant of $\mathbb{N}$ is the number of **k**-points that lie within the Brillouin zone.

Grids generated by the Moreno-Soler method are *Generalized Regular* (GR) grids. GR grids have never been widely adopted because they require a search over many supercells to select the cell that 1) maximizes the distance between points and 2) has the fewest irreducible **k**-points. These searches tend to be time-consuming due to the combinatoric explosion in the total number of possible supercells as shown in Fig. 5.5.

The following study compares MP, GR, and simultaneously commensurate (SC) (a set of grids often used by the Cluster Expansion community) to determine how the type of **k**-point grid affects the errors. The data also

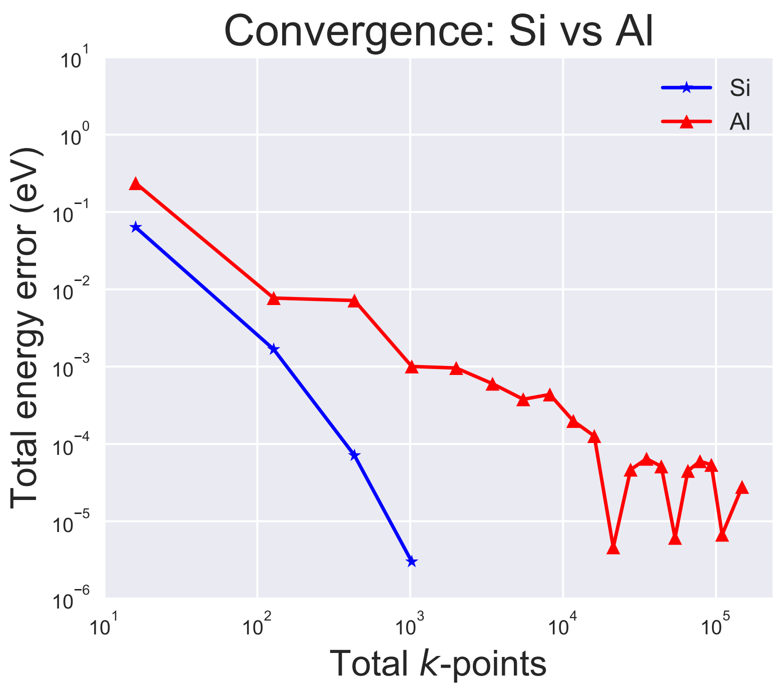


Figure 4.1 Total energy error vs. total number of **k**-points for the cases of silicon and aluminum. Silicon does not have a Fermi surface so there is no discontinuity in the occupied bands; convergence is super-exponential or $\mathscr{O}(e^n)$ where n is the number of **k**-points. (See the discussion of example 1 in Ref. [2].) In contrast, the total energy of aluminum converges very slowly, and the convergence is quite erratic. For typical target accuracies in the total energy, around 10^{-3} eV/atom, metals require 10–50 times more **k**-points than semiconductors.

provides a more detailed study of how **k**-point convergence behaves within DFT for metals. This information allows DFT users to know which type of **k**-point grid will work best and how dense the **k**-point sampling should be to achieve a target accuracy.

For this article, I performed all the calculations and analysis found in the paper as well as wrote several portions of the paper. Jeremy Jorgensen wrote the majority of the paper. All the authors contributed ideas regarding the data analysis and helped make the subject understandable to the reader.

The following article is reproduced with permission. A license can be found in Appendix E.

Computational Materials Science 153 (2018) 424–430

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Efficiency of Generalized Regular $\mathbf{k}$-point grids

Wiley S. Morgan*, Jeremy J. Jorgensen, Bret C. Hess, Gus L.W. Hart

Department of Physics and Astronomy, Brigham Young University, Provo, UT 84602, USA

A B S T R A C T

Most DFT practitioners use regular grids (Monkhorst-Pack, MP) for integrations in the Brillouin zone. Although regular grids are the natural choice and easy to generate, more general grids whose generating vectors are not merely integer divisions of the reciprocal lattice vectors, are usually more efficient (Wisesa et al., 2016). We demonstrate the efficiency of *generalized regular* (GR) grids compared to Monkhorst-Pack (MP) and *simultaneously commensurate* (SC) grids. In the case of metals, for total energy accuracies of one meV/atom, GR grids are 60% faster on average than MP grids and 20% faster than SC grids. GR grids also have greater freedom in choosing the $\mathbf{k}$-point density, enabling the practitioner to achieve a target accuracy with the minimum computational cost.

1. Introduction

High throughput materials design has become an effective route to material discovery with many successes already documented [2–31]. The creation of large material databases is the first step in high throughput approaches [32–51]. Computationally expensive electronic structure calculations generate the data for the databases and limit the extent to which data analysis tools, such as machine learning, can be applied. Increasing the speed of these calculations has the potential to significantly increase the size of these databases and the impact of material predictions.

Most electronic structure codes perform numerical integrals over the first Brillouin zone, which converge extremely slowly in the case of metals. Dense sampling of the Brillouin zone, required for high accuracy, is computationally expensive, especially when implementing hybrid functionals or perturbative expansions in density functional theory (DFT) [52]. High accuracy is important because the energies of competing phases are often similar and even small errors can affect the prediction of stable materials.

Methods for $\mathbf{k}$-point selection have not changed much since Monkhorst and Pack published their influential paper over 40 years ago [53]. Their method was quickly accepted by the community due to its simplicity and ability to generalize previous methods [54,55]. Sampling methods that improve upon Monkhorst-Pack (MP) grids have been far less prevalent [1,56–58].

In this paper, we compare the $\mathbf{k}$-point selection method promoted by Wisesa, McGill, and Mueller [1] (WMM) to the standard MP grids and to another common method in the alloy community, which we refer to as *simultaneously commensurate* (SC) grids. This paper serves to reinforce and quantify the claims made by WMM, as applied to calculations typically used for alloys and for some high-throughput studies.

2. Background

Over the past 40 years, only a few $\mathbf{k}$-point selection methods have been proposed in the literature [1,53–57]. Many of these so-called special point methods have focused on selecting points that accurately determined the mean value of a periodic function defined over the Brillouin zone because the integral of a periodic function over one period is simply its mean value. Other factors that have been considered in developing special point methods are selection of grids with a consistent density in each direction and full exploitation of symmetry.

Baldereschi introduced the *mean-value point* of the Brillouin zone [54], the first special point method. In this approach, the periodic function to be integrated is written as a Fourier expansion:

$$f(\mathbf{k}) = \sum_{n=0}^{\infty} c_n e^{i\mathbf{k}\cdot\mathbf{R}_n}, \tag{1}$$

where $\mathbf{k}$ is the wavevector, c_n is the n-th expansion coefficient, and the sum is over over all lattice points $\mathbf{R}_n$. Baldereschi noted that the integral of $f(\mathbf{k})$ within the first Brillouin zone (i.e., over one period of $f(\mathbf{k})$), is proportional to the leading coefficient, c_0, in the Fourier expansion,

$$\int_{\mathrm{BZ}} f(\mathbf{k})\,d\mathbf{k} = \frac{(2\pi)^3}{\Omega}c_0, \tag{2}$$

where Ω is the volume of the reciprocal cell. He replaced the analytic

* Corresponding author.
E-mail address: wiley.s.morgan@gmail.com (W.S. Morgan).

https://doi.org/10.1016/j.commatsci.2018.06.031
Received 18 April 2018; Received in revised form 21 June 2018; Accepted 22 June 2018
Available online 20 July 2018

integral of the periodic function with a numeric integral (sum over j in Eq. (3))—equivalent in the limit of infinite sampling points—and replaced the periodic function with its infinite Fourier expansion (sum over n in Eq. (3)):

$$\begin{aligned}\int_{\mathrm{BZ}} f(\mathbf{k}) &= \sum_{j=0}^{\infty} w_j f(\mathbf{k}_j)\\ &= \sum_{j=0}^{\infty} w_j \sum_{n=0}^{\infty} c_n e^{i\mathbf{k}_j\cdot\mathbf{R}_n}\\ &= \sum_{j=0}^{\infty} w_j (c_0 + c_1 e^{i\mathbf{k}_j\cdot\mathbf{R}_1} + \dots)\\ &= \sum_{j=0}^{\infty} w_j c_0 + \sum_{j=0}^{\infty} w_j c_1 e^{i\mathbf{k}_j\cdot\mathbf{R}_1} + \dots,\end{aligned} \tag{3}$$

where w_j is the integration weight of the j-th $\mathbf{k}$-point. In the final step of Eq. (3), each term (sum over j) is a numeric integral of the n-th basis function in the Fourier expansion of $f(\mathbf{k})$ (denoted as I_n in what follows). Baldereschi's method selected $\mathbf{k}$-points so that the leading terms after c_0 integrate to zero:

$$\begin{aligned}\int_{\mathrm{BZ}} f(\mathbf{k}) &= \sum_{j=0}^{\infty} w_j c_0 + \sum_{j=0}^{\infty} w_j c_1 e^{i\mathbf{k}_j\cdot\mathbf{R}_1} +\\ &\quad \sum_{j=0}^{\infty} w_j c_2 e^{i\mathbf{k}_j\cdot\mathbf{R}_2} + \dots\\ &= I_0 + \cancelto{0}{I_1} + \cancelto{0}{I_2} + \mathcal{O}(I_3),\\ &\approx c_0 \sum_j w_j.\end{aligned}$$

This is an accurate approximation when the Fourier coefficients converge rapidly to zero, as is the case with insulators and semiconductors. Baldereschi's approximation is ineffective for metals because the integral over the occupied parts of the band structure has discontinuities, and the Fourier series converges very slowly.

Chadi and Cohen extended the mean-value point by introducing *sets* of $\mathbf{k}$-points whose weighted sum eliminated the contribution of a greater number of leading basis functions [55]. Their sets of $\mathbf{k}$-points could be made as dense as desired.

The most popular $\mathbf{k}$-point selection method was created by Monkhorst and Pack [53] (MP). They established a grid of points that generalized both the mean-value point of Baldereschi and its extension by Chadi and Cohen and which was equivalent to points used by Janak et al. [59] MP grids are given by the relation

$$\mathbf{k}_{prs} = u_p\mathbf{b}_1 + u_r\mathbf{b}_2 + u_s\mathbf{b}_3 \tag{4}$$

where $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$ are the reciprocal lattice vectors, $u_p = (2p{-}q{-}1)/2q$ for $p = 1, 2, \dots, q$, and q an integer that determines the grid density. The same relation holds for u_r and u_s. In other words, the generating vectors of MP grids are simply integer divisions of the reciprocal lattice vectors.

Froyen generalized the MP points, which he called *Fourier quadrature points*, by eliminating the restriction that the vectors that defined the grid be parallel to the reciprocal lattice vectors [56]. However, he did require the grid to be commensurate with the reciprocal lattice and to have the full point-group symmetry of the crystal.

Moreno and Soler [57] introduced the idea of searching for $\mathbf{k}$-point grids with the fewest points for a given length cutoff—a parameter that characterized the quality of the grid and was closely related to the $\mathbf{k}$-point density. Their method constructs superlattices of the real-space primitive lattice. The *dual* of the superlattice vectors form the $\mathbf{k}$-point grid generating vectors. By selecting superlattices that maximize the minimum distance between lattice points (i.e., by choosing fcc-like superlattices), they obtain $\mathbf{k}$-point grids that are bcc-like. Grids that are bcc-like have the smallest integration errors at a given $\mathbf{k}$-point density. (This is evident in Fig. 6.) Moreno and Soler further improved Brillouin zone sampling by finding the offset of the origin that maximized the symmetry reduction of the grid.

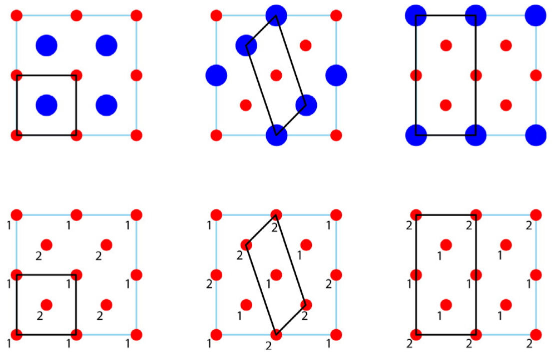


Fig. 1. In order to isolate the effect of the Brillouin zone shape and size on total energy error when comparing crystal structures of different shapes and sizes (top row), the energy of supercells (bottom row) crystallographically equivalent to single element, primitive cells were compared. The total energy per atom should be the same for all equivalent cells.

In their recent paper, WMM point out that the lack of popularity of Moreno and Soler's approach is due to the computational expense of calculating many Froyen grids and searching for the ones with the highest symmetry reduction. They used the term *Generalized Monkhorst-Pack* (GMP) grids to refer to Froyen grids with the highest symmetry reduction for a given $\mathbf{k}$-point density. We refer to these grids as *Generalized Regular* (GR) grids since they are simply generalizations of the regular grids used in finite element, finite difference, and related methods. WMM precalculated the grids, and stored the ones with the

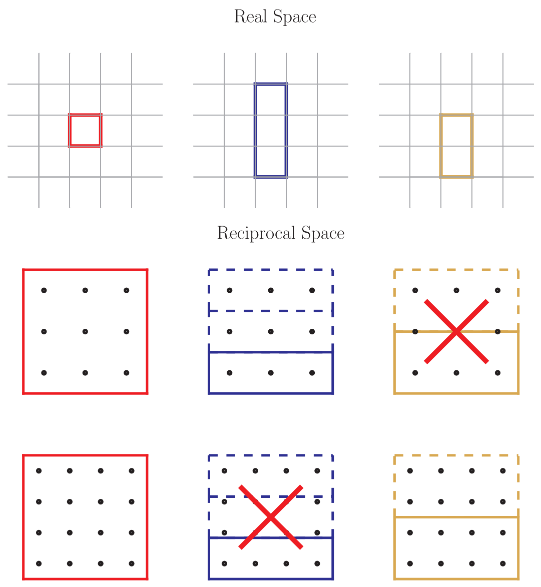


Fig. 2. An example of simultaneously commensurate grids. The cells for each crystal are shown in both real and reciprocal space. In reciprocal space, we include two $\mathbf{k}$-point grids of different density. Simultaneously commensurate grids eliminate systematic $\mathbf{k}$-point error (between two commensurate structures) by using the same grid for both the parent cell (red cell) and the supercells (yellow and blue). However, some grids may not be allowed (crossed out) for a given supercell because they are incommensurate with the reciprocal cell. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

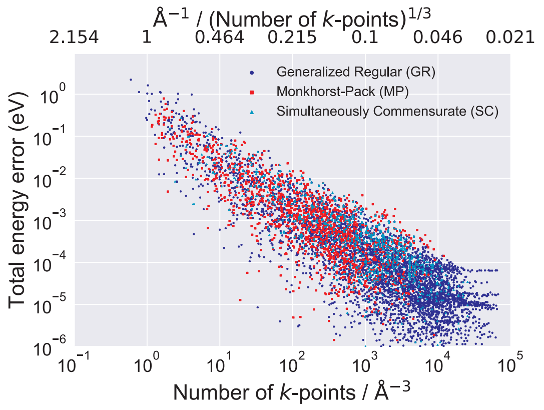


Fig. 3. Total energy convergence by grid type. Note that the size of the convergence envelope gets bigger with higher **k**-point densities.

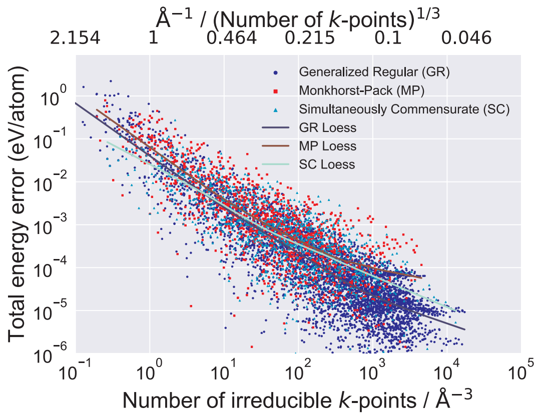


Fig. 4. Total energy convergence with respect to the irreducible **k**-point density. By looking at the irreducible **k**-point density the efficiency of the different grids can be distinguished. Loess smoothing was also employed to determine the average efficiency of the grids.

highest symmetry reduction in a database that can be accessed via an internet request.

3. Methods

We compare the total energy errors of MP, SC, and GR grids for different **k**-point densities over calculations of nine different elements (all of which are metallic), many cell shapes, and cell sizes from 1 to 14 atoms. In total we compare errors across more than 7000 total energy calculations. One **k**-point grid is considered superior to another if it requires a smaller irreducible **k**-point density to reach a specific accuracy target (for example 10^{-3} eV/atom). The method that requires the smallest irreducible **k**-point density is the one we regard as best suited for high throughput and machine learning applications.

To isolate error arising from **k**-point integration, the different cells were crystallographically equivalent to single element, primitive cells, as illustrated in Fig. 1. We did this to study how **k**-point error depends on the Brillouin zone shape and size; this is an important consideration in high-throughput studies where total energy differences between competing phases are critical.

The grid types we compared were MP grids (generated by AFLOW's algorithm [32]), SC grids [56] (examples of SC grids can be found in Fig. 2, details of SC grid generation can be found in the appendix), and GR grids (generated by querying WMM's **k**-point server) [60]. We ran DFT calculations using the Vienna Ab-initio Simulation Package (VASP) [61–64] on nine monoatomic systems—Al, Pd, Cu, W, V, K, Ti, Y, and Re—using PAW PBE pseudopotentials [65,66]. The supercells of cubic systems varied between 1 and 11 atoms per cell, while the hexagonal close packed (HCP) systems had 2–14 atoms per cell. We used VASP 4.6 for all calculations[1]. For MP and SC grids the target number of **k**-points extended from 10 to 10,000 unreduced **k**-points The range of **k**-points for GR grids was 4–150,000 **k**-points[2].

The converged total energy, the energy taken as the error-free "solution" in our energy convergence comparisons, was the calculation with the highest **k**-point density for each system. Because MP and SC grids are difficult to generate at comparable densities, GR grids were used to generate the converged total energy.

[1] For SC grids an independent k-point folding algorithm was used due to an occasional bug in VASP 4.6's folding algorithm. This bug has been fixed in version 6 of VASP.

[2] Calculations with MP grids with more than about 10,000 unreduced points were not used. These calculations were problematic due to a number of problems, including memory constraints and errors during the k-point folding.

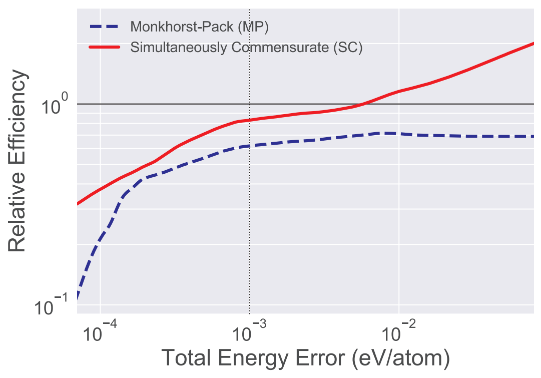


Fig. 5. Relative grid efficiency. Along the y-axis are the ratios of the MP and SC efficiencies compared to the GR grid efficiency (black horizontal line at 10^0). Total energy error (per atom) is plotted along the x-axis and decreases to the left. MP and SC grids are generally less efficient than GR grids: at a target accuracy of 1 meV/atom, MP grids are 60%, and SC grids are 20%, less efficient.

4. Results

In Fig. 3, we show the convergence for the MP, SC, and GR grids with respect to the **k**-point density, i.e., **k**-points/Å^{-3}. The first thing to note is the large spread in the convergence. This spread reduces the reliability of high-throughput databases and is perhaps higher than one might expect. Note that the size of the total energy convergence envelope (variance) gets bigger with increasing **k**-point densities. Additionally it can be seen that each method has the same variance at all **k**-point densities.

In order to quantify the efficiency of GR grids relative to SC and MP grids, we studied the rate of energy convergence with respect to the irreducible **k**-point density, i.e., the number of *irreducible* **k**-points divided by the volume of the reciprocal cell in Å^{-3} (shown in Fig. 4). Given the amount of scatter in the plot, we performed loess regression to create trend lines for each grid type.

The efficiency of a **k**-point grid is proportional to the irreducible **k**-point density required to reach a given accuracy. Comparisons of efficiencies were made by taking the ratio of the GR trend line to the SC and MP trend lines of Fig. 5. At accuracies higher than 5 meV/atom, GR grids are more efficient (averaged over many structures) than MP and SC grids. As an example, at a target accuracy of 1 meV/atom, the GR grids are 20%

more efficient than SC grids and 60% more efficient than MP grids.

It should be noted that in both Figs. 4 and 5 that MP grids appear to perform worse at higher densities than at lower densities. Our statistical analysis has indicated that this behavior is not statistically significant and likely results from data scarcity for MP grids at high densities. We believe that with sufficient data for MP grids at these densities the trend line would continue to run roughly parallel to the GR line across all densities. However, due to the computational expense of generating MP grids at such densities we have been unable to demonstrate this.

5. Discussion

The erratic convergence of total energy for metals is attributed principally to the Fermi surface. Integrating over the occupied portions of the band structure is equivalent to integrating a discontinuous band structure over the Brillouin zone; the rapid, monotonic convergence observed for insulators and semiconductors is lost because of the surface of discontinuities.

It is perhaps surprising how much the error varies at a given **k**-point density. The implication is that, when generating databases of total energies, relatively high **k**-point densities will be required for accurate comparisons. For example, in Fig. 4, **k**-point densities as low as 10s of **k**-points/Å^{-3} achieve 10^{-3} eV/atom error for some structures, but to be certain that *all* structures are converged to the same accuracy densities as high as 5000 **k**-points/Å^{-3} are necessary. Given the spread in the data we recommend that a target density of 5000 **k**-points/Å^{-3} be used to reliably achieve accuracies of 10^{-3} eV/atom for metals. However, should another accuracy be desired, one can simply follow the top edge of the distribution of points in Fig. 3 to the desired accuracy and read off the corresponding density.

For reference: a **k**-point density of 5000 **k**-points/Å^{-3} corresponds to a linear **k**-point density of 0.058 Å^{-1} (common input scheme for CASTEP or newer versions of VASP, `KSPACING` in the `INCAR` file). This is equivalent to the following Monkhorst-Pack grids or "**k**-point per reciprocal atom" (KPPRA) settings for a few pure elements:

Element	Cell divisions	KPPRA
W	43 × 43 × 43	80,000
Cu	48 × 48 × 48	110,500
Al	43 × 43 × 43	80,000
K	26 × 26 × 26	17,500
Ti (2 atoms, hcp)	41 × 41 × 21	18,000

Likely these high numbers will be surprising to most DFT practitioners—indeed, the current authors found them so—but this is the message of Fig. 4 if one wants to be fully converged in all cases, and not just on average. The large scatter in the errors for a given density imposes this large penalty on the practitioner who wishes to have fully converged calculations. The need for high densities when using a regular grid for DFT calculations of metallic systems highlights the need for development of adaptive techniques that can mitigate the deleterious effects of a discontinuous integrand (i.e., the existence of a Fermi surface.)

In our tests of GR grids, we also observed large spread in the energy convergence of insulators, rather than the typical monotonic convergence observed for MP grids. This happens because GR grids are not restricted to a single Bravais lattice type. Grids of different lattice types will have different packing fractions and thus converge at different rates. Fig. 6 shows the energy convergence rate of primitive silicon for three Bravais lattice types; the convergence is monotonic for each type. As expected, body-centered cubic grids have the fastest convergence. This is because bcc lattices have the highest packing fraction when Fourier transformed (becoming fcc). If grids of multiple Bravais lattice types are used, as happens for GR grids obtained by querying WMM's **k**-point server, spread in the energy convergence is introduced. To demonstrate that erratic convergence for metals is not merely due to mixing grids of multiple Bravais lattice types, we include Fig. 7. The figure also demonstrates that the grid type, i.e., bcc, fcc, or sc, has no effect on the convergence.

6. Conclusion

GR grids are not intrinsically better than SC or MP grids—that is, they do not converge more rapidly as a function of **k**-point density. They are more efficient because they typically have better symmetry reduction than MP or SC grids, reducing the computational effort required for GR grids. Also, with GR grids one may increase the **k**-point density in smaller increments because the set of possible grids (and thus **k**-point densities) is larger than the sets of possible grids for SC and MP.

Our tests over more than 7000 structures of varying cell sizes, shapes, and **k**-point densities demonstrate how erratic **k**-point convergence is for metals, and how wide the variance can be at a given **k**-point density, and how this variance grows with increasing **k**-point densities. These facts should be considered when generating computational materials databases since greater errors may result from not

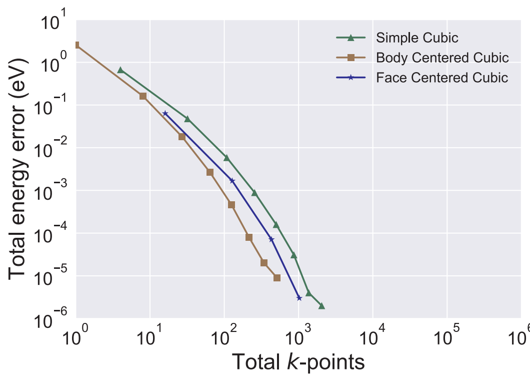


Fig. 6. Convergence for silicon by Bravais lattice type of the **k**-point grid. The energy convergence remains smooth for GR grids as long as the grid is of a single Bravais lattice type. Otherwise, some spread in the energy convergence, similar to that observed for metals, is introduced.

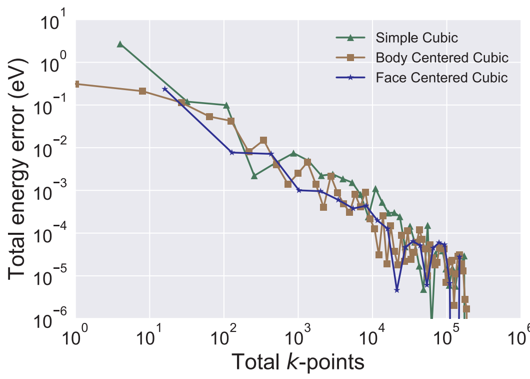


Fig. 7. (Color online) Convergence of aluminum by Bravais lattice type of the **k**-point grid. Jaggedness and spread in the energy convergence remains for GR grids even after separating the grids by Bravais lattice type.

using enough **k**-points for a target accuracy. Using GR grids for nonmetals may result in unexpected scatter; when smooth convergence is desired, we advise that GR grids of a single Bravais lattice type be utilized.

Acknowledgments

The authors are grateful to Shane Reese who helped with the loess regression and statistical analysis of the data shown Fig. 4. The authors are grateful to Tim Mueller and Georg Kresse for helpful discussions. This work was supported under: ONR (MURI N00014-13-1-0635).

The raw and processed data required to reproduce these findings are available to download from https://github.com/wsmorgan/GR_Grid_Comparisons.

Appendix A. Simultaneously commensurate grid construction

A *simultaneously commensurate* (SC) grid is useful for calculating formation enthalpies when the target structure is a derivative superstructure of a parent structure. (Obviously this is a convenient method when computing enthalpies for cluster expansion studies because the training structures are superstructures of the parent.) When SC grids are used, the absolute convergence with respect to **k**-point density is not faster than for other grids but the *relative convergence* can be faster because of error cancellation—both the parent structure and the derivative superstructure have exactly the same grid. The idea is illustrated in Fig. 8. In panel (a) we divide up the reciprocal unit cell of the parent lattice (red[3] parallelogram) into a uniform grid of **k**-points (blue points). We then place the same grid from the parent cell on the supercell, as in panel (b). If we have chosen a SC grid, it is clearly periodic for the supercell as well as the parent. Only those grids that are commensurate with both the parent cell and supercell can be used to integrate both cells. Fig. 9 shows an example of an incommensurate grid. When the grid of the parent cell is placed over the reciprocal cell of the supercell, the grid is *not periodic*—translations of the supercell (dotted lines) are sampled differently by the grid.

For our crystals that have cubic parent cells, an initial set of commensurate bcc, fcc, and sc grids were generated. A subset of those grids that were commensurate with each supercell were used to do calculations of the various crystal structures. For hexagonal crystals, a similar procedure was followed except only hexagonal grids were used.

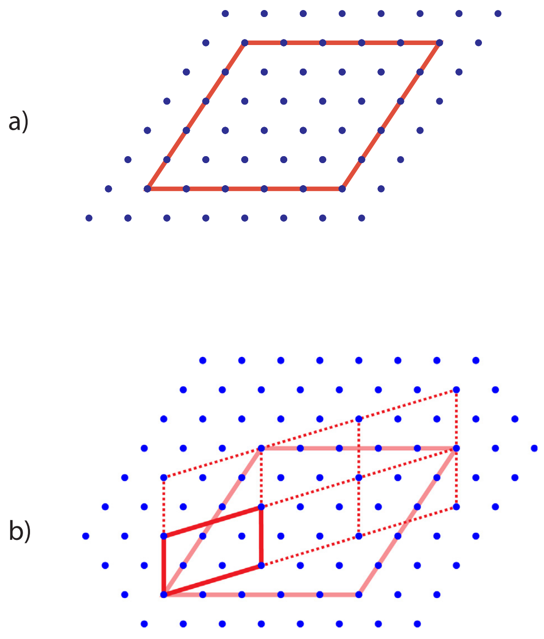


Fig. 8. (Color online) An example of a SC grid. Panel a): the selected grid (blue points) is commensurate with the reciprocal cell (red parallelogram) of the parent cell. Panel b): it can be seen that the grid is also commensurate with the reciprocal unit cell of the supercell (smaller parallelogram). (Dotted lines indicate translations of the supercell.) The **k**-point grid is the same in each translation of the supercell.

[3] For interpretation of color in Fig. 8, the reader is referred to the web version of this article.

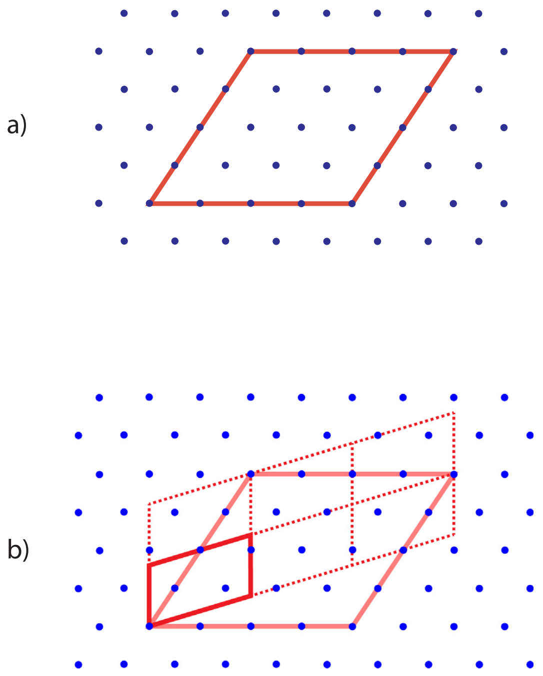


Fig. 9. (Color online) An example of a SC grid (blue dots) that is not commensurate with both the reciprocal unit cell of the parent cell and the reciprocal unit cell of the supercell. Panel a): the chosen grid is commensurate with the parent cell (red parallelogram). Panel b): the **k**-point grid is not commensurate with the supercell—it is not periodic; the grid is not the same in each translational copy (dotted lines) of the supercell.

Appendix B. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.commatsci.2018.06.031.

References

[1] P. Wisesa, K.A. McGill, T. Mueller, Phys. Rev. B 93 (2016) 155109.
[2] J. Greeley, T.F. Jaramillo, J. Bonde, I. Chorkendorff, J.K. Nørskov, Nat. Mater. 5 (2006) 909.
[3] R. Gautier, X. Zhang, L. Hu, L. Yu, Y. Lin, T.O. Sunde, D. Chon, K.R. Poeppelmeier, A. Zunger, Nat. Chem. 7 (2015) 308.
[4] A.O. Oliynyk, A. Mar, Acc. Chem. Res. (2017).
[5] H. Chen, G. Hautier, A. Jain, C. Moore, B. Kang, R. Doe, L. Wu, Y. Zhu, Y. Tang, G. Ceder, Chem. Mater. 24 (2012) 2009.
[6] G. Hautier, A. Jain, S.P. Ong, B. Kang, C. Moore, R. Doe, G. Ceder, Chem. Mater. 23 (2011) 3495.
[7] C. Jahne, C. Neef, C. Koo, H.-P. Meyer, R. Klingeler, J. Mater. Chem. A 1 (2013) 2856.
[8] T. Moot, O. Isayev, R.W. Call, S.M. McCullough, M. Ze-maitis, R. Lopez, J.F. Cahoon, A. Tropsha, Mater. Discov. 6 (2016) 9.
[9] U. Aydemir, J.-H. Pohls, H. Zhu, G. Hautier, S. Bajaj, Z.M. Gibbs, W. Chen, G. Li, S. Ohno, D. Broberg, et al., J. Mater. Chem. A 4 (2016) 2461.
[10] H. Zhu, G. Hautier, U. Aydemir, Z.M. Gibbs, G. Li, S. Bajaj, J.-H. Pohls, D. Broberg, W. Chen, A. Jain, et al., J. Mater. Chem. C 3 (2015) 10554.
[11] W. Chen, J.-H. Pohls, G. Hautier, D. Broberg, S. Bajaj, U. Aydemir, Z.M. Gibbs, H. Zhu, M. Asta, G.J. Snyder, et al., J. Mater. Chem. C 4 (2016) 4414.
[12] G. Ceder, Y.-M. Chiang, D. Sadoway, M. Aydinol, Y.-I. Jang, B. Huang, Nature 392 (1998) 694.
[13] F. Yan, X. Zhang, G.Y. Yonggang, L. Yu, A. Nagaraja, T.O. Mason, A. Zunger, Nat. Commun. 6 (2015).
[14] D. Bende, F.R. Wagner, O. Sichevych, Y. Grin, Angew. Chem. 129 (2017) 1333.
[15] A. Mannodi-Kanakkithodi, A. Chandrasekaran, C. Kim, T.D. Huan, G. Pilania, V. Botu, R. Ramprasad, Mater. Today (2017).
[16] S. Sanvito, C. Oses, J. Xue, A. Tiwari, M. Zic, T. Archer, P. Tozman, M. Venkatesan, M. Coey, S. Curtarolo, Sci. Adv. 3 (2017) e1602241.
[17] H. Yaghoobnejad Asl, A. Choudhury, Chem. Mater. 28 (2016) 5029.
[18] G. Hautier, A. Miglio, G. Ceder, G.-M. Rignanese, X. Gonze, Nat. Commun. 4 (2013) 2292.
[19] A. Bhatia, G. Hautier, T. Nilgianskul, A. Miglio, J. Sun, H.J. Kim, K.H. Kim, S. Chen, G.-M. Rignanese, X. Gonze, et al., Chem. Mater. 28 (2015) 30.
[20] G.H. Johannesson, T. Bligaard, A.V. Ruban, H.L. Skriver, K.W. Jacobsen, J.K. Nørskov, Phys. Rev. Lett. 88 (2002) 255506.
[21] D.P. Stucke, V.H. Crespi, Nano Lett. 3 (2003) 1183.
[22] S. Curtarolo, D. Morgan, G. Ceder, Calphad 29 (2005) 163.
[23] S.F. Matar, I. Baraille, M. Subramanian, Chem. Phys. 355 (2009) 43.
[24] G. Ceder, G. Hautier, A. Jain, S.P. Ong, MRS Bull. 36 (2011) 185.
[25] A.N. Sokolov, S. Atahan-Evrenk, R. Mondal, H.B. Akker-man, R.S. Sanchez-Carrera, S. Granados-Focil, J. Schrier, S.C. Mannsfeld, A.P. Zoombelt, Z. Bao, et al., Nat. Commun. 2 (2011) 437.
[26] Z.W. Ulissi, M.T. Tang, J. Xiao, X. Liu, D.A. Torelli, M. Karamad, K. Cummins, C. Hahn, N.S. Lewis, T.F. Jaramillo, et al., ACS Catal. 7 (2017) 6600.
[27] O. Levy, R.V. Chepulskii, G.L. Hart, S. Curtarolo, JACS 132 (2009) 833.
[28] X. Ma, G. Hautier, A. Jain, R. Doe, G. Ceder, J. Electrochem. Soc. 160 (2013) A279.
[29] K. Yang, W. Setyawan, S. Wang, M.B. Nardelli, S. Curtarolo, Nat. Mater. 11 (2012) 614.
[30] H. Chen, G. Hautier, G. Ceder, JACS 134 (2012) 19619.
[31] S. Kirklin, B. Meredig, C. Wolverton, Adv. Energy Mater. 3 (2013) 252.
[32] S. Curtarolo, W. Setyawan, G.L. Hart, M. Jahnatek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, et al., Comput. Mater. Sci. 58 (2012) 218.
[33] J.E. Saal, S. Kirklin, M. Aykol, B. Meredig, C. Wolverton, JOM 65 (2013) 1501.
[34] A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, et al., APL Mater. 1 (2013) 011002.
[35] S.L. Digabel, C. Tribes, C. Audet, NOMAD User Guide. Tech. Rep. G-2009-37, Les cahiers du GERAD, Quebec, Canada, 2009.
[36] D.D. Landis, J.S. Hummelshøj, S. Nestorov, J. Greeley, M. Dulak, T. Bligaard, J.K. N0rskov, K.W. Jacobsen, Comput. Sci. Eng. 14 (2012) 51.
[37] J. Hachmann, R. Olivares-Amaya, S. Atahan-Evrenk, C. Amador-Bedolla, R.S. Sanchez-Carrera, A. GoldParker, L. Vogt, A.M. Brockway, A. Aspuru-Guzik, J. Phys. Chem. Lett. 2 (2011) 2241.
[38] J.S. Hummelshøj, F. Abild-Pedersen, F. Studt, T. Bligaard, J.K. Nørskov, Angew. Chem. 124 (2012) 278.
[39] M. De Jong, W. Chen, H. Geerlings, M. Asta, K.A. Persson, Sci. Data 2 (2015).
[40] M. De Jong, W. Chen, T. Angsten, A. Jain, R. Notestine, A. Gamst, M. Sluiter, C.K. Ande, S. Van Der Zwaag, J.J. Plata, et al., Sci. Data 2 (2015) 150009.
[41] L. Cheng, R.S. Assary, X. Qu, A. Jain, S.P. Ong, N.N. Rajput, K. Persson, L.A. Curtiss, J. Phys. Chem. Lett. 6 (2015) 283.
[42] R. Gomez-Bombarelli, J. Aguilera-Iparraguirre, T.D. Hirzel, D. Duvenaud, D. Maclaurin, M.A. Blood-Forsythe, H.S. Chae, M. Einzinger, D.-G. Ha, T. Wu, et al., Nat. Mater. 15 (2016) 1120.
[43] E.M. Chan, Chem. Soc. Rev. 44 (2015) 1653.
[44] T. Tada, S. Takemoto, S. Matsuishi, H. Hosono, Inorg. Chem. 53 (2014) 10347.

[45] G. Pilania, C. Wang, X. Jiang, S. Rajasekaran, R. Ramprasad, Sci. Rep. 3 (2013).
[46] J. Yan, P. Gorai, B. Ortiz, S. Miller, S.A. Barnett, T. Mason, V. Stevanovic, E.S. Toberer, Energy Environ. Sci. 8 (2015) 983.
[47] R. Ramakrishnan, P.O. Dral, M. Rupp, O.A. Von Lilienfeld, Sci. Data 1 (2014) 140022.
[48] J. Hachmann, R. Olivares-Amaya, A. Jinich, A.L. Ap-pleton, M.A. Blood-Forsythe, L.R. Seress, C. Roman-Salgado, K. Trepte, S. Atahan-Evrenk, S. Er, et al., Energy Environ. Sci. 7 (2014) 698.
[49] L.-C. Lin, A.H. Berger, R.L. Martin, J. Kim, J.A. Swisher, K. Jariwala, C.H. Rycroft, A.S. Bhown, M.W. Deem, M. Haranczyk, et al., Nat. Mater. 11 (2012) 633.
[50] R. Armiento, B. Kozinsky, G. Hautier, M. Fornari, G. Ceder, Phys. Rev. B 89 (2014) 134103.
[51] O. Senkov, J. Miller, D. Miracle, C. Woodward, Nat. Commun. 6 (2015).
[52] K. Berland, C. Persson, Comput. Mater. Sci. 134 (2017) 17.
[53] H.J. Monkhorst, J.D. Pack, Phys. Rev. B 13 (1976) 5188.
[54] A. Baldereschi, Phys. Rev. B 7 (1973) 5212.
[55] D.J. Chadi, M.L. Cohen, Phys. Rev. B 8 (1973) 5747.
[56] S. Froyen, Phys. Rev. B 39 (1989) 3168.
[57] J. Moreno, J.M. Soler, Phys. Rev. B 45 (1992) 13891.
[58] E. Cances, V. Ehrlacher, D. Gontier, A. Levitt, D. Lombardi, Available from: < arXiv:1805.07144 > .
[59] J. Janak, Computational Methods in Band Theory, Springer, 1971, pp. 323–339.
[60] http://muellergroup.jhu.edu/K-Points.html.
[61] G. Kresse, J. Hafner, Phys. Rev. B 47 (1993) 558.
[62] G. Kresse, J. Furthmuller, Comput. Mater. Sci. 6 (1996) 15.
[63] G. Kresse, J. Hafner, Phys. Rev. B 49 (1994) 14251.
[64] G. Kresse, J. Furthmuller, Phys. Rev. B 54 (1996) 11169.
[65] P.E. Blochl, Phys. Rev. B 50 (1994) 17953.
[66] G. Kresse, D. Joubert, Phys. Rev. B 59 (1999) 1758.

CHAPTER 5

Generating k-point Grids

As discussed in Chapter 4, general regular (GR) **k**-point grids offer many advantages over traditional Monkhorst-Pack (MP) grids. However, GR grids have yet to be widely adopted by the DFT community.

Recently Wisesa, McGill, and Mueller [74] rectified this by creating a **k**-point server containing precalculated grids that have high symmetry reduction. These grids can be retrieved via an Internet request. However, the requirement of an Internet query, which cannot be performed in typical supercomputer environments, makes them difficult to use in some cases. For GR grids to be used widely throughout the DFT community they will need to be generated at runtime. This section details just such an algorithm which can generate GR grids on the fly.

Section 5.1 details an efficient **k**-point folding algorithm that allows for many grids to be folded and compared in seconds. The algorithm relies heavily on concepts used in the enumeration algorithm of the `enumlib` code [1]. The algorithm has been implemented in an open-source code available at https://github.com/msg-byu/kgridGen. The algorithm has also been incorporated in version 6 of the VASP code [75]. This algorithm has been submitted for publication.

Section 5.2 contains an algorithm that generates GR grids and selects the optimal grid for a crystal structure. This algorithm has been implemented in the package `GRkgridgen` , implemented at https://github.com/msg-byu/GRkgridgen, and has also been submitted for publication. This algorithm has not, as of this writing, been incorporated into VASP.

Both of these algorithms will have significant impact on the DFT community and will likely be used for many years.

In the first half of this chapter, Section 5.1, Dr. Gus Hart designed and implemented most of the algorithm and wrote the majority of the paper from which the text of Section 5.1 was taken. Jeremy Jorgensen helped write the algorithm and implemented tests. Dr. Rodney Forcade helped with the proof of the algorithm which can be found in Appendix A. I helped test the code and implemented code that corrects the space group of the crystals and improved the code's numerical stability.

In Section 5.2, I was responsible for the algorithm's design, its implementation, the initial testing, and writing the paper from which the text of Section 5.2 was taken. Parker Hamilton and John Christensen helped improve the algorithm's efficiency and contributed more tests. Dr. Rodney Forcade contributed general formulas that were used in the algorithms described in 5.2.1. Dr. Branton Campbell contributed invaluable ideas regarding canonical bases for crystal systems and symmetry-preserving offsets.

5.1 FOLDING K-POINTS

In most DFT codes the setup and symmetry reduction of the **k**-point grid only takes a few seconds. The motivation for an improved algorithm, is two-fold: 1) eliminate the probability of incorrect symmetry reduction due to finite precision errors (the danger of these increases as the density of the integration grid increases), and 2) enable the automatic grid-generation algorithm described in 5.2.

5.1.1 Generating Grids

Every uniform sampling of a Brillouin zone can be expressed through the simple integer relationship

$$\mathbb{R} = \mathbb{K}\mathbb{N}. \tag{5.1}$$

In Eq. (5.1), $\mathbb{R}$, $\mathbb{K}$, and $\mathbb{N}$ are 3×3 matrices; the columns of $\mathbb{R}$ are the reciprocal lattice vectors, the columns of $\mathbb{K}$ are the **k**-point grid generating vectors. $\mathbb{N}$ is a transformation matrix from the vectors of $\mathbb{K}$ to the vectors of $\mathbb{R}$. Monkhorst-Pack grids (regular grids) occur when $\mathbb{N}$ is an integer, diagonal matrix. More generally, when $\mathbb{N}$ is an invertible, integer matrix, one obtains generalized regular grids. A two dimensional example of the concept is illustrated in Fig. 5.1. Throughout the rest of this dissertation let the infinite lattice of points defined by integer linear combinations of $\mathbb{R}$ be represented by R, and the lattice of points defined by $\mathbb{K}$ as K.

With no loss of generality, a new basis for the grid K can be chosen (a different, but equivalent, $\mathbb{K}$) so that $\mathbb{N}$ is in the canonical Hermite normal form (HNF), $\mathbb{H}$ ($\mathbb{H}$ is the canonical HNF of $\mathbb{N}$), subject to the constraints:

$$\begin{aligned} \mathbb{H} &= \begin{pmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{pmatrix} \\ & a, c, f > 0 \\ & b \geq 0, \quad b < c \\ & d, e \geq 0, \quad d, e < f \end{aligned} \tag{5.2}$$

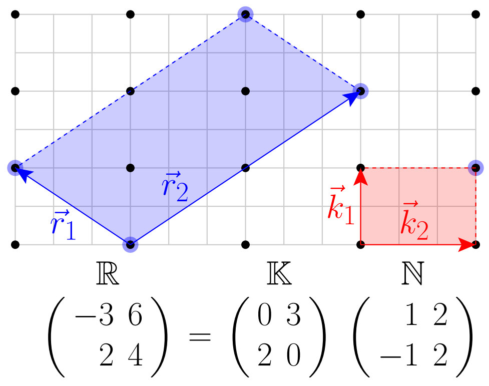

$\vec{r}_1$
$\vec{r}_2$
$\vec{k}_1$
$\vec{k}_2$
$\mathbb{R}$
$\mathbb{K}$
$\mathbb{N}$
$\begin{pmatrix} -3 & 6 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 3 \\ 2 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ -1 & 2 \end{pmatrix}$

Figure 5.1 Example of the integer relationship between reciprocal lattice vectors $\mathbb{R}$ and the grid generating vectors $\mathbb{K}$. In the picture, the grid generating vectors, $\vec{\kappa}_1$ and $\vec{\kappa}_2$, the columns of $\mathbb{K}$, define a lattice of points, four of which are inside the unit cell of $\mathbb{R}$. Note that in the most general case, the relationship between the two lattices, $\mathbb{N}$ need not be diagonal (as it is for Monkhorst-Pack [3] **k**-point grids.)

The integration grid is the set of points of the lattice K that lie inside one unit cell of the reciprocal lattice R. We refer to this finite subset of K as K_α (See Fig. 5.1; black dots are K, dots inside the blue parallelogram comprise

[1]If two points are translationally distinct, their difference cannot be an integer linear combination of the *reciprocal* cell vectors; that is, $\vec{k}_i - \vec{k}_j \neq n\vec{r}_1 + m\vec{r}_2 + \ell\vec{r}_3$, for all integer values n, m, ℓ. ($\vec{r}_i$ are the columns of $\mathbb{R}$.)

K_α.) The number of points that lie within one unit cell of R is given by $|\det(\mathbb{N})| = n = a \times c \times f$. The grid can then be constructed by generating a set of n translationally distinct[1] points of the lattice K. These points can be generated by taking integer linear combinations of the basis of K:

$$\vec{k} = p\vec{\kappa}_1 + q\vec{\kappa}_2 + r\vec{\kappa}_3, \tag{5.3}$$

where p, q, and r are nonnegative integers such that

$$\begin{aligned} 0 \le p &< a \\ 0 \le q &< c \\ 0 \le r &< f, \end{aligned}$$

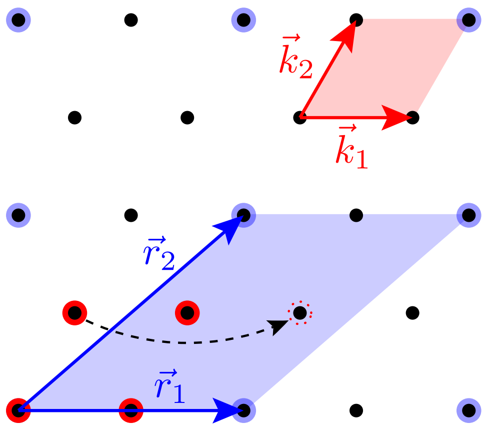
and $\vec{\kappa}_1$, $\vec{\kappa}_2$, and $\vec{\kappa}_3$ are the columns of $\mathbb{K}$. The n points generated this way will not generally lie inside the same unit cell, but they can be translated into the same cell by expressing them in "lattice coordinates" (fractions of the columns of $\mathbb{R}$, instead of Cartesian coordinates) and then taking the coordinates modulo 1 so that they all lie within the range $[0,1)$. This is illustrated by the dashed arrow in Fig. 5.2.

The points shown in Fig 5.2 expressed as fractions of the lattice vectors of R are:

$$\begin{aligned} \vec{k}_1 &= (0,0) \\ \vec{k}_2 &= (\tfrac{1}{2},0) \\ \vec{k}_3 &= (-\tfrac{1}{4},\tfrac{1}{2}) \xrightarrow{\text{mod } 1} (\tfrac{3}{4},\tfrac{1}{2}) \\ \vec{k}_4 &= (\tfrac{1}{4},\tfrac{1}{2}). \end{aligned}$$

Initially, k_3 is not in the same unit cell as the other three points; its first coordinate is not between 0 and 1. Taking the first coordinate modulo 1, moves k_3 to an equivalent position in the same unit cell as the other three points.

This first part of the algorithm generates n translationally distinct points within the first unit cell of R. (In practice it is not necessary to translate all the points into the first unit cell. It is, however, good practice to translate all the points into the first Brillouin zone, and a method for doing this is discussed in Sec. 5.1.3.)



$$\begin{aligned} \vec{\kappa}_1 &= (1,0) \\ \vec{\kappa}_2 &= \left(\tfrac{1}{2},\tfrac{\sqrt{3}}{2}\right) \end{aligned} \qquad \mathbb{K} = (\vec{\kappa}_1,\vec{\kappa}_2)$$

$$\mathbb{R} = \begin{pmatrix} 2 & 2 \\ 0 & \sqrt{3} \end{pmatrix} \qquad \mathbb{N} = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}$$

Figure 5.2 An example of generating the points of K (black lattice) that lie within one unit cell (blue parallelogram) of the lattice R (blue lattice). The lattice K is generated by the basis $\{\vec{\kappa}_1,\vec{\kappa}_2\}$ (columns of $\mathbb{K}$). The four points of K_α are generated by $\vec{k} = m_1\vec{\kappa}_1 + m_2\vec{\kappa}_2$, where $0 \le m_1 < 2$, $0 \le m_2 < 2$. Note that the upper limits of m_1 and m_2 are the diagonals of $\mathbb{N}$ when it is expressed in HNF.

5.1.2 Symmetry Reduction of the Grid

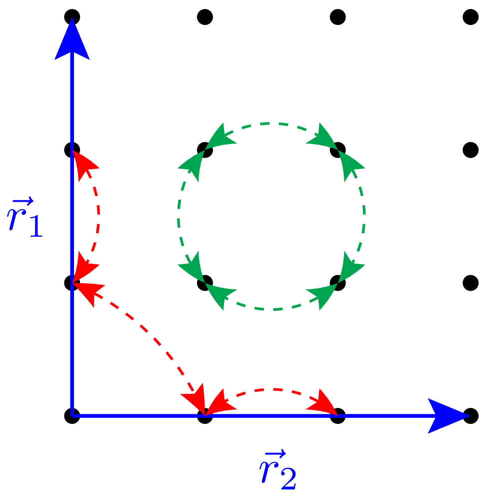


Figure 5.3 An example of symmetry reducing a grid. The reciprocal unit cell is a square. This example assumes that the wavefunctions have square symmetry as well (the D_4 group, 8 operations). The example grid is a 3×3 sampling of the reciprocal unit cell. The point at $(0,0)$ is not equivalent to any of the other eight points. There are two sets of equivalent points, each set with 4 points in the orbit, connected by red and green arrows, respectively. The points marked by red arrows are equivalent under horizontal, diagonal, and vertical reflections about the center of the square. The green-marked points are equivalent by 90° rotations. Thus the nine points are reduced (or "folded") into *three* symmetrically distinct points.

At this point the point group symmetries of the crystal can be exploited to reduce the number of **k**-points where the energy eigenvalues need to be evaluated. The grid is reduced by applying the symmetry operations of the crystal[2] to each point in the grid. For example, in Fig. 5.3, the points connected by green arrows will be mapped onto one another by successive 90° rotations. These four symmetrically equivalent points lie on a 4-fold "orbit" (as do the points marked by the red arrows). The point at the origin maps onto itself under all symmetry operations and has an orbit of length 1 (the length of an orbit is equivalent to the number of points in the orbit).

For a grid containing N_k points and a group (of rotation and reflection symmetries) with N_G operations, a naive algorithm for identifying the symmetrically equivalent points and counting the length of each orbit would be as follows: For each point ($\mathcal{O}(N_k)$ loop), compare all rotations of that point (a loop of $\mathcal{O}(|G|)$) to all other points (another $\mathcal{O}(N_k)$ loop) to find a match; for a total computational complexity of $\mathcal{O}(N_k^2 N_G)$ (where N_G is the number of rotation and reflection symmetries). In pseudocode, this algorithm would be the following:

[2]In our implementation of the algorithm time reversal symmetry, which implies that E(**k**) = E(-**k**), is enforced even for crystals which lack inversion symmetry.

Algorithm 1

```
uniqueCount $\longleftarrow 0$
    for each $k_i \in K_\alpha$
        orbitLength; $\longleftarrow 0$
        unique $\longleftarrow$ true
        for each $k_j \in K_\alpha$
            for each $g \in G$
                if $k_j = g \cdot k_i$
                    orbitLength; ++
                    unique $\longleftarrow$ false
            end
        end
        if (unique)
            uniqueCount ++
            add $k_i$ to list of unique points
```

N_G will never be larger than 48, but N_k may be in the hundreds of thousands for extremely dense grids, so the N_k^2 complexity of this approach is undesirable. But using group theory concepts (see Appendix B for details), we can construct a hash table for the points that reduces the complexity from $\mathcal{O}(N_k^2 N_G)$ to $\mathcal{O}(N_k N_G)$ by eliminating the k_j loop in the above algorithm. The hash table makes a one-to-one association between the ordinal counter (i.e., the index) for each point and its coordinates.

The three coefficients p, q, r in Eq. (5.3) can be conceptualized as the three "digits" of a 3-digit mixed-radix number pqr or as the three numerals shown on an odometer with three wheels. The ranges of the values are $0 \leq p < d_1$, $0 \leq q < d_2$, and $0 \leq r < d_3$, where d_1, d_2, d_3 are the "sizes" of the wheels, or in other words, the *base* of each digit. Then the mixed-radix number is converted to base 10 as

$$x = p \cdot d_2 \cdot d_1 + q \cdot d_1 + r. \tag{5.4}$$

The total number of possible readings of the odometer is $d_3 \cdot d_2 \cdot d_1$. So it must be the case that the number of **k**-points in the cell is $n = d_3 \cdot d_2 \cdot d_1$ (the d_i are not necessarily equal to the diagonals of the HNF). Each reading on the odometer is a distinct point of the n points that are contained in the reciprocal cell. Via Eq. (5.4) it is simple to convert a point given in "lattice coordinates" as (p, q, r) to a base-10 number x. The concept of the hash table is to use this base-10 representation as the index in the hash table. Without the hash table, comparing two points is an $\mathcal{O}(N_k)$ search because one point must be compared to every other point in the list to check for equivalency. But with the hash function, mapping (p, q, r) to x only requires a single comparison.

It is not generally the case that the coefficients p, q, r for every interior point of the unit cell obey conditions:

$$0 \leq p < d_1, \quad 0 \leq q < d_2, \quad 0 \leq r < d_3 \tag{5.5}$$

(Fig. 5.2 shows an example where the interior points do not meet these conditions.) These conditions hold only for a certain choice of basis. That basis is found by transforming the matrix $\mathbb{N}$ in Eq. (5.1) into its Smith Normal Form (SNF), $\mathbb{D} = \mathbb{A}\mathbb{N}\mathbb{B}$. By elementary row and column operations, represented by unimodular matrices $\mathbb{A}$ and $\mathbb{B}$, it is possible to transform $\mathbb{N}$ into a diagonal matrix $\mathbb{D}$, where each diagonal element divides the ones below it: $d_{11}|d_{22}|d_{33}$, and $d_{11} \cdot d_{22} \cdot d_{33} = n = |\mathbb{N}|$ (the notation $i|j$ means that j is divisible by i). As explained in Appendix B, when $\mathbb{N}$ is expressed in SNF and the interior points of the reciprocal cell are expressed as linear combinations of the grid generating vectors $\mathbb{K}$, then the coordinates (coefficients) of the interior points will obey Eq. 5.5. When these conditions are met, the hashing algorithm discussed above (in particular, Eq. 5.4) becomes possible. This enables the $\mathcal{O}(N_k)$ algorithm, given here as Algorithm 2:

Algorithm 2

```
uniqueCount $\longleftarrow 0$
hashTable[:]  $\longleftarrow 0$
First[:]  $\longleftarrow 0$
Wt[:]  $\longleftarrow 0$
    for each $k_i \in K$
        indx $\longleftarrow \mathbb{K}^{-1}\mathbb{A}\mathbb{D} \cdot k_i$
        if hashTable[indx] $\neq 0$ cycle #this
        #point and all its symmetry
        #equivalent points has already been
        #indexed
        uniqueCount++
        hashtable[indx] $\longleftarrow$ uniqueCount
        First[uniqueCount] $\longleftarrow$ i
        Wt[uniqueCount] $\longleftarrow$ 1
        # Now mark all equivalent points
        for each $g \in G$
            $k_{\text{rot}} \longleftarrow g \cdot k_i$
            indx $\longleftarrow \mathbb{K}^{-1}\mathbb{A}\mathbb{D} \cdot k_{\text{rot}}$
            if hashtable[indx] == 0
                hashtable[indx] $\longleftarrow$ uniqueCount
                Wt[uniqueCount]++
```

At the end of Algorithm 2, the array `Wt` will be a list of the weights for each symmetrically unique point, and the index of each unique point in K will be stored in the array `First`.

5.1.3 Moving Points Into the First Brillouin Zone

For accurate DFT calculations, it is best if the energy eigenvalues (electronic bands) are evaluated at **k**-points inside the first Brillouin zone; translating grid points into the Brillouin zone is the final step in the **k**-point folding algorithm. (In principle, the electronic structure should be the same in every unit cell, but numerically the periodicity of the electronic bands is only approximate, becoming less accurate for **k**-points in unit cells farther from the origin.)

The first Brillouin zone of the reciprocal lattice is a Voronoi cell centered at the origin—all **k**-points in the first Brillouin zone are closer to the origin than to any other lattice point. Conceptually, an algorithm for translating a **k**-point of the integration grid into the first zone merely requires one to look at all translationally equivalent "cousins" of the **k**-point and select the one closest to the origin. The number of translationally equivalent cousins is, of course, countably infinite so in practice, the set of cousins must be limited to those near the origin.

The key idea to selecting a set of cousins that contains the closest cousin to the origin is illustrated in two-dimensions in Fig. 5.4. In three-dimensions, if the basis vectors of the reciprocal unit cell are as short as possible (the Minkowski-reduced basis), then the eight unit cells that all share a vertex at the origin *must* contain every point that is closer to the origin than any other point. In other words, the boundary of this union of eight cells is guaranteed to circumscribe the first Brillouin zone (i.e., the Voronoi cell containing the origin). A proof of this "8 cells" conjecture is given in Appendix A. The steps for moving **k**-points into the Brillouin zone are as follows:

1. Minkowski-reduce the reciprocal unit cell [76] (i.e., find the basis with the shortest basis vectors.)
2. For each **k**-point in the reduced grid, find the translation-equivalent cousin in each of the eight unit cells that have a vertex at the origin.
3. From these eight cousins, select the one closest to the origin.

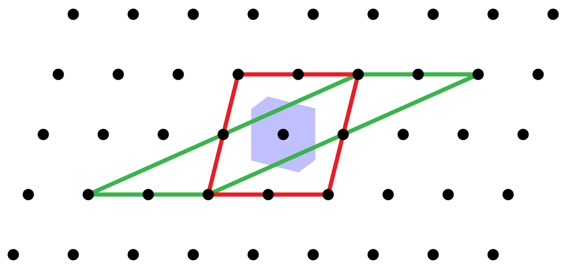


Figure 5.4 In 2D, the first Brillion zone, shown in blue, is a subset of the union of 4 basis cells around the origin, shown in red, when the basis vectors are chosen to be as short as possible (the so-called Minkowski basis). If the basis is not Minkowski reduced, regions of the Brillouin zone may lie outside the union of the 4 basis cells, which is depicted by the cell in green.

5.2 GENERATING K-POINT GRIDS ON THE FLY

5.2.1 Algorithm Details

The main difficulty in generating GR grids is that the number of distinct supercells grows rapidly with the volume factor (the determinant of $\mathbb{H}$[3]). To optimize the **k**-point folding

[3]$\mathbb{H}$ is the canonical HNF of $\mathbb{N}$ and has the same determinant.

efficiency, the **k**-point grid should have the same symmetry as the parent cell. The number of supercells that preserve the symmetry of the parent is always significantly smaller than the number of possible supercells (except in the case of triclinic lattices) as can be seen in Fig. 5.5. If one can generate only those supercells that preserve the symmetry of the parent the computational burden is drastically reduced. The steps required to identify a crystal, find the symmetry-preserving grids and filter them for the ideal grid are described in detail below.

Generating Symmetry-Preserving Supercells

To generate only the symmetry-preserving supercells, the integer matrices in Eq. 4.2 are restricted to be Hermite Normal Form (HNF) $\mathbb{H}$, as defined in Section 5.1.1. Throughout this Section of the dissertation the following notation will be used: $\mathbb{A} = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ for the parent lattice and $\mathbb{C} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ for a supercell such that $\mathbb{C} = \mathbb{A}\mathbb{H}$. When the lattice symmetries are applied to $\mathbb{A}$, another set of basis vectors $\mathbb{A}'$ is generated

$$\mathbb{A}' = \mathbf{g}\mathbb{A} \tag{5.6}$$

(where $\mathbf{g}$ is an element of the point group). Because $\mathbb{A}$ and $\mathbb{A}'$ are related by a symmetry operation of the lattice, they both represent the same lattice and are related by an integer matrix

$$\begin{aligned} \mathbb{A}' &= \mathbb{A}\mathbb{X} \\ \mathbb{A}\mathbb{X} &= \mathbf{g}\mathbb{A} \\ \mathbb{X} &= \mathbb{A}^{-1}\mathbf{g}\mathbb{A} \end{aligned} \tag{5.7}$$

where $\mathbb{X}$ is an integer matrix with determinant ± 1. Similarly, if a supercell $\mathbb{C}$ has the same symmetry as $\mathbb{A}$ then all the symmetries of $\mathbb{A}$ will map $\mathbb{C}$ to another basis $\mathbb{C}'$ that will be related to $\mathbb{C}$ by a unimodular transformation

$$\begin{aligned} \mathbb{C}' &= \mathbf{g}\mathbb{C} \;\forall\, \mathbf{g} \in \mathbf{G} \\ \mathbb{C}\mathbb{M} &= \mathbf{g}\mathbb{C} \\ \mathbb{M} &= \mathbb{C}^{-1}\mathbf{g}\mathbb{C} \end{aligned} \tag{5.8}$$

where $\mathbf{G}$ is the set of generators of the point group of $\mathbb{A}$ and $\mathbb{M}$ is an integer matrix. Using Eqs. (5.7) and (5.8), it is possible to define restrictions on the entries of $\mathbb{H}$:

$$\mathbb{M} = \mathbb{H}^{-1}\mathbb{X}\mathbb{H} \tag{5.9}$$

In other words $\mathbb{H}$ must transform $\mathbb{X}$ such that $\mathbb{M}$ retains integer entries. Equation (5.9) yields the following system of linear equations

$$\begin{aligned}
\alpha_1 &= \frac{bx_{12} + dx_{13}}{a} \\
\alpha_2 &= \frac{cx_{12} + ex_{13}}{a} \\
\alpha_3 &= \frac{fx_{13}}{a} \\
\beta_1 &= \frac{-bx_{11} + ax_{21} - b\alpha_1 + bx_{22} + dx_{23}}{c} \\
\beta_2 &= \frac{-b\alpha_2 + ex_{23}}{c} \\
\beta_3 &= \frac{-b\alpha_3 + cx_{23}}{c} \\
f &= \frac{\alpha_4}{c} \\
\gamma_1 &= \frac{ax_{31} + bx_{32} + dx_{33} - e\beta_1 - d\alpha_1 - dx_{11}}{f} \\
\gamma_2 &= \frac{-ex_{22} + cx_{32} + ex_{33} - e\beta_2 - d\alpha_2}{f} \\
n &= a \cdot c \cdot f
\end{aligned} \tag{5.10}$$

where x_i are the entries of $\mathbb{X}$, n is the determinant of $\mathbb{H}$ and α_i, β_i, and γ_i are arbitrary names for the expressions used for convenience. $\mathbb{H}$ will generate a supercell that preserves the symmetries of $\mathbb{A}$ when α_1, α_2, α_3, α_4, β_1, β_2, β_3, γ_1, and γ_2 are all integers for each generator in $\mathbf{G}$. Even though the solutions to (5.10) have no closed form, we may use them to build an algorithm that generates $\mathbb{H}$ matrices that preserve the lattice symmetries.

The specific form of $\mathbb{X}$ depends on the basis chosen for the parent lattice, the solutions to (5.10), and resulting algorithms, will differ depending on the basis. For example, if a base-centered orthorhombic lattice is constructed with the basis

$$\mathbb{A}_1 = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 3 \end{pmatrix} \tag{5.11}$$

then (5.10) would reduce to (each equation has three outputs because the base centered orthorhombic point-group has three generators):

$$\begin{aligned} \alpha_1 &= \left(0,\ 0,\ -\tfrac{b}{a}\right) \\ \alpha_2 &= \left(0,\ 0,\ -\tfrac{c}{a}\right) \\ \alpha_3 = \beta_3 &= \left(0,\ 0,\ 0\right) \\ \beta_1 &= \left(0,\ 0,\ \tfrac{-a-b\alpha_1}{c}\right) \\ \beta_2 &= \left(0,\ 0,\ \tfrac{b}{a}\right) \\ \gamma_1 &= \left(0,\ \tfrac{2d}{f},\ \tfrac{-d-d\alpha_1-e\beta_1}{f}\right) \\ \gamma_2 &= \left(0,\ \tfrac{2e}{f},\ \tfrac{-e-d\alpha_2-e\beta_2}{f}\right) \end{aligned} \tag{5.12}$$

All the equations in (5.12) must be simultaneously satisfied for the generated $\mathbb{H}$'s to preserve the symmetries of $\mathbb{A}_1$. Alternatively the basis

$$\mathbb{A}_2 = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 1 & -2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \tag{5.13}$$

could be used to construct the same lattice. When basis $\mathbb{A}_2$ is chosen, the relations in (5.10) become:

$$\begin{aligned} \alpha_1 = \alpha_2 = \alpha_3 = \beta_2 = \beta_3 &= \left(0,\ 0,\ 0\right) \\ \beta_1 &= \left(0,\ 0,\ \tfrac{a+2b}{c}\right) \\ \gamma_1 &= \left(0,\ \tfrac{2d}{f},\ \tfrac{-e\beta_1}{f}\right) \\ \gamma_2 &= \left(0,\ \tfrac{2e}{f},\ -\tfrac{2e}{f}\right) \end{aligned} \tag{5.14}$$

Note the stark difference between the relationships derived from $\mathbb{A}_1$ and $\mathbb{A}_2$. $\mathbb{A}_2$ results in fewer equations to check, however, $\mathbb{A}_1$ gives relationships between a and b, and a and c separately resulting in a faster search since many combinations can be skipped early in the search. By taking care in selecting a basis for each lattice, one can find an efficient set of conditions for generating the supercells of that basis. For a complete list of all bases used in the `GRkgridgen` , and the resulting relationships from Eqs. 5.10 can be found in Appendix C.

Niggli Reduction

Choosing a basis for each type of lattice presents a problem; there are an infinite number of lattice basis choices. The number of bases is substantially reduced by recognizing that any given symmetry-preserving HNF, $\mathbb{H}^{\text{sp}}$, will work for *every* lattice of the same symmetry. The sensitivity of the representation of the point group $\mathbb{X}$ on the chosen basis requires a set of representative bases that goes beyond the 14 Bravais lattices. Such a set was constructed by Niggli [77, 78, 78–80], who identified 44 distinct bases. Any given basis of a crystal can be classified as one of these 44 cases by reducing it to the Niggli canonical form and then comparing the lengths of the basis vectors and the angles between them. If two nominally different lattices reduce to the same Niggli case, then the two lattices are "equivalent" and have the same symmetries and the same set of $\mathbb{H}^{\text{sp}}$s.

Niggli reduction allows for the user's basis to be mapped to a basis which has convenient solutions to Eqs. (5.10). The strategy is to define the $\mathbb{H}^{\text{sp}}$s in the selected basis, then generate the supercells for the selected basis and transform them to the $\mathbb{H}$'s for the Niggli reduced basis, $\mathbb{H}^{\text{sp}}_R$. Once the $\mathbb{H}^{\text{sp}}_R$'s have been determined, they can be applied directly to the user's reduced basis to create a symmetry-

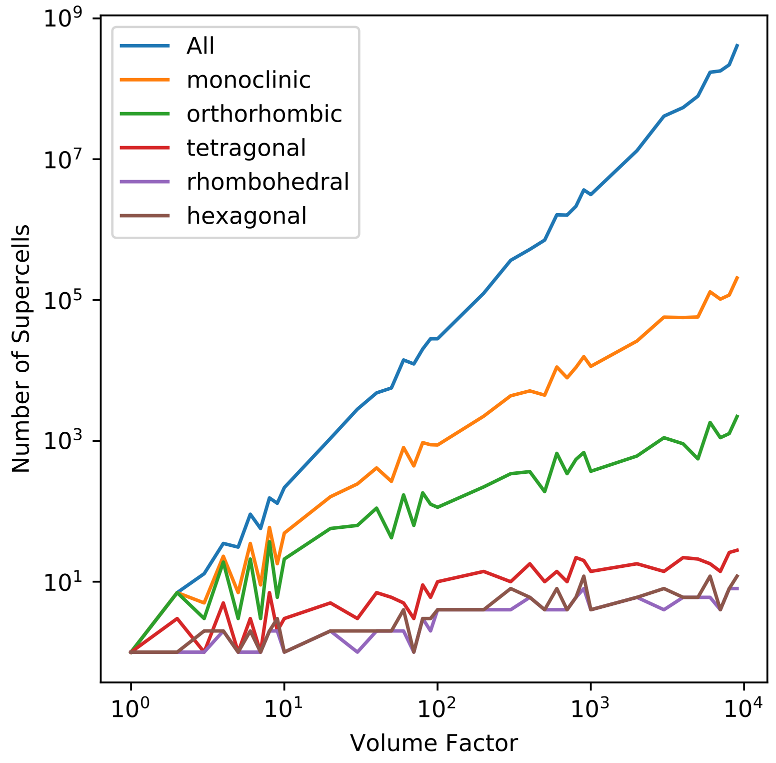


Figure 5.5 The number of supercells that preserve the symmetry of the parent cell at various volume factors. The total number of supercells that exist is also displayed for comparison. Cubic cells were omitted since they have at most one symmetry-preserving supercell at a given volume factor.

preserving supercell of the user's parent cell and thus define an efficient **k**-point grid at the specified density.

Grid Selection

At a given volume factor (i.e., number of **k**-points), the integer relations in Eq. (5.10) will yield multiple supercells for most lattices, a 2D example of these supercells is provided in Fig. 5.6(a). It is then necessary to select one which defines the best **k**-point grid. This is done by transforming each symmetry-preserving supercell to its corresponding **k**-point grid generating vectors as in Eq. 4.3; see Fig. 5.6(b). This set of grids is then searched for one that has optimal properties—a uniform distribution of points and the best symmetry reduction. To ensure the grid generating vectors are as short as possible we perform Minkowski reduction [76], then sort the grids by the length of their shortest vector.

The most uniform grids will have the longest shortest vector. The grids are filtered so that none with a packing fraction of less then 0.3 are considered. Each of the uniform grids is then symmetry reduced [81] in order to determine which has the fewest irreducible

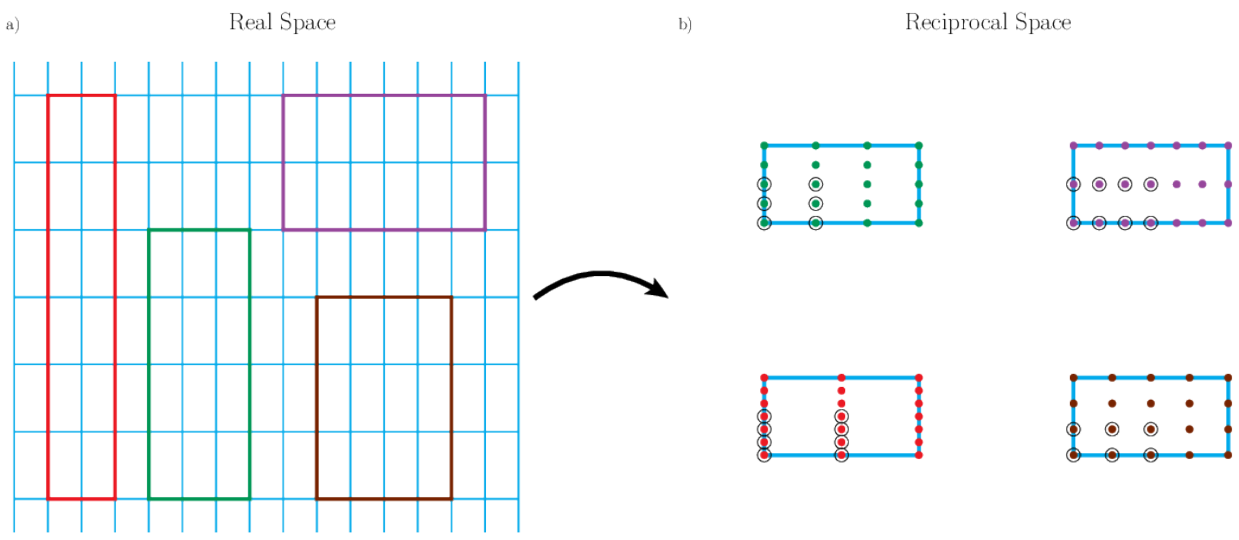


Figure 5.6 A 2D example of symmetry-preserving supercells and the **k**-point grids that they would generate for a rectangular lattice. a) contains four symmetry-preserving supercells of the primitive cell, shown in blue, with a volume factor of 12. In b) the primitive cell, blue cells, and the supercells have been mapped to reciprocal space and the grids generated from each supercell have been placed in separate cells. The color of the grid points matches the color of the generating supercell. The circled points are the irreducible **k**-points of each grid.

k-points. Table 5.1 shows the length of the shortest vector and number of irreducible **k**-points for the grids in Fig. 5.6(b). The grids are sorted first by the length of their shortest vector (eliminating the green and red grids) then by the number of irreducible **k**-points such that the ideal grid appears at the top of the table, i.e., the grid generated by the brown supercell in Fig. 5.6(a).

It is also possible to offset the **k**-point grid from the origin to improve the grid's efficiency. The origin is not symmetrically equivalent to any other point in the grid. Including an offset moves the point off the origin making it possible for the point to be mapped to other points in the grid, decreasing the number of irreducible **k**-points. Different grids have different symmetry-preserving offsets that should be tested. For example, both simple cubic and face-centered cubic (fcc) grids have one possible offset that preserves the full symmetry of the cell, $(\frac{1}{2},\frac{1}{2},\frac{1}{2})$ (expressed as fractions of the grid generating vectors), while a body-centered-cubic lattice has no symmetry-preserving offsets[4], and simple tetragonal has three symmetry-preserving offsets. (For a full list of the symmetry-preserving offsets by lattice type see Appendix D.) The grid that has the fewest **k**-points with a given offset is selected.

Not every volume factor will have a symmetry-preserving grid that is uniform. To ensure that a symmetry-preserving grid is found, it is necessary to include multiple

[4]For some lattices no symmetry-preserving offsets exist. In these cases using an offset that does not preserve the full symmetry can be beneficial. For example, a body centered cubic system with an offset of $(0,0,\frac{1}{2})$ can sometimes offer better folding than the same grid with no offset.

grid	shortest vector length	number of irreducible **k**-points
brown	0.16667	6
purple	0.166667	8
green	0.125	6
red	0.08333	8

Table 5.1 Properties (length of shortest vector and number of irreducible **k**-points) of the grids in Fig 5.6

volume factors in the search. The number of additional volume factors to search depends on the lattice type; in general, the search should continue until multiple candidate grids have been found. The best grid is then selected from these candidates.

Method Summary

The algorithm can be summarized in the following steps:

1. Identify the Niggli reduced cell of the user's structure.
2. Generate the symmetry-preserving HNFs for the canonical form of the Niggli cell.
3. Map the resulting supercells to the original lattice using the Niggli-reduced basis as an intermediary.
4. Convert the supercells into **k**-point grid generating vectors.
5. Perform Minkowski reduction on the grid generating vectors.
6. Sort the grid generating vectors by the length of their shortest vector.
7. Select the grids whose shortest vectors have the longest lengths.
8. Use the symmetry group to reduce the selected grids to find the one with the fewest irreducible **k**-points.

5.2.2 Testing the Algorithm

To test the above algorithm, the **k**-point grids it generates, GR_{auto}, were compared to those generated by the **k**-point sever [74], GR_{server} in two ways. First, **k**-point grids were generated by GR_{server} and GR_{auto} over a range of **k**-point densities for over 100 crystal lattices. These lattices were constructed for nine monoatomic systems—Al, Pd, Cu, W, V, K, Ti, Y, and Re—with supercells for the cubic systems having between 1–11 atoms per cell and supercells for the hexagonal close packed systems having between 2–14 atoms per cell, the same set used in Chapter 4. Additional structures were selected from AFLOW [6]. All tests were conducted without offsetting the grids from Γ, the origin. The resulting ratio of irreducible **k**-points to total **k**-points was then plotted for each grid. Six representative examples of the results are shown in Fig. 5.7. These tests show that GR_{auto} grids should be very close in performance to GR_{server} grids. Additionally, the tests show that convergence toward the ideal folding ratio is rapid for all lattice types.

The second test compared the total energy errors of MP (generated by AFLOW), GR_{auto} and GR_{server} grids in the same manner, and using the same methods, as done in the study of *GR* grids found in Chapter 4. A brief review of that method is provided here.

DFT calculations were performed using the Vienna Ab-initio Simulation Package

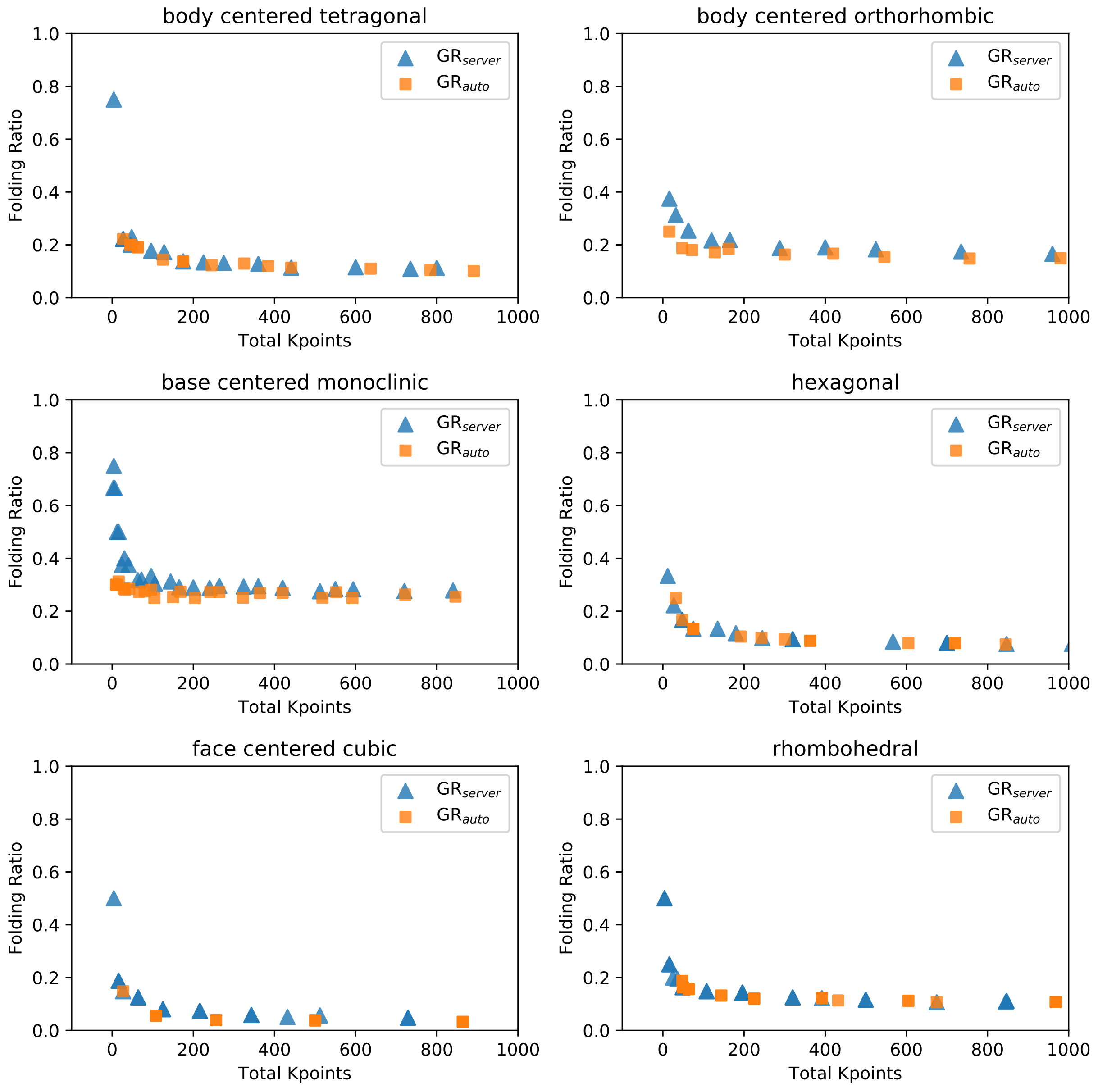


Figure 5.7 A comparison of the GR_{auto} and GR_{server} **k**-point grids. For each grid th number of irreducible **k**-points was divided by the total number of **k**-points. This shows that both sets of grids offer similar folding at a given **k**-point density and will have similar efficiencies.

4.6 (VASP 4.6) [75, 82–84] on the nine monoatomic systems mentioned above using PAW PBE pseudopotentials. [85,86] In order to isolate the errors from **k**-point integration, the different cells were crystallographically equivalent to single element cells. For MP grids, the target number of **k**-points ranged from 10-10,000 unreduced **k**-points, for GR_{server} grids the range was 4–240,000 unreduced **k**-points, and for GR_{auto} the range was 8 to 415,000 unreduced **k**-points. In total, we compared errors across more than 7000 total energy calculations. The energy taken as the error-free "solution" in our comparisons was the calculation with the highest **k**-point density for each system. The total error convergence with respect to the **k**-point density is shown in Fig. 5.8. The total error convergence with respect to the number of irreducible **k**-points were compared using loess regression, see Fig. 5.9. Ratios of these trend lines were then taken to determine the efficiency of each grid relative to the GR_{server} grids (see Fig. 5.10).

From Figs. 5.9 and 5.10, it can be seen that GR_{auto} grids are up to $\sim$10% more efficient and at worst $\sim$5% less efficient than GR_{server} grids. Both sets of grids outperform MP grids by $\sim$60% at an accuracy target of 1 meV/atom. The runtime for the algorithm to generate GR_{auto} grids at a **k**-point density of 5000 (dense enough to acheive 1 meV/atom accuracy) was $\sim$4 seconds on average.

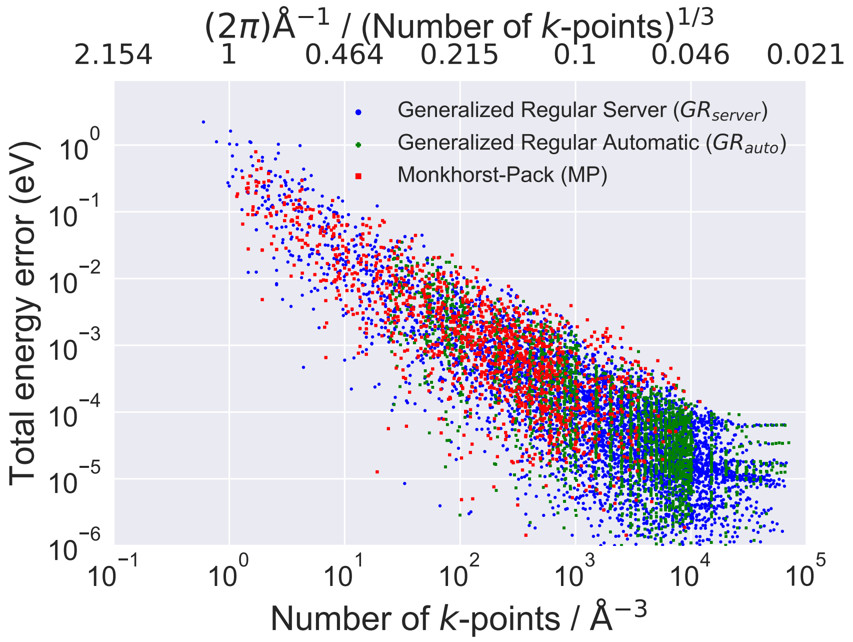


Figure 5.8 The total energy convergence with respect to total **k**-point density for MP, GR_{auto} and GR_{server} grids. The top axis shows the linear **k**-point spacing with a factor of 2π included as part of the transformation to reciprocal space. This differs from the linear **k**-point spacing usually used as input in DFT codes by a factor of 2π, i.e., to get the spacing used as input in codes divide the values here by 2π.

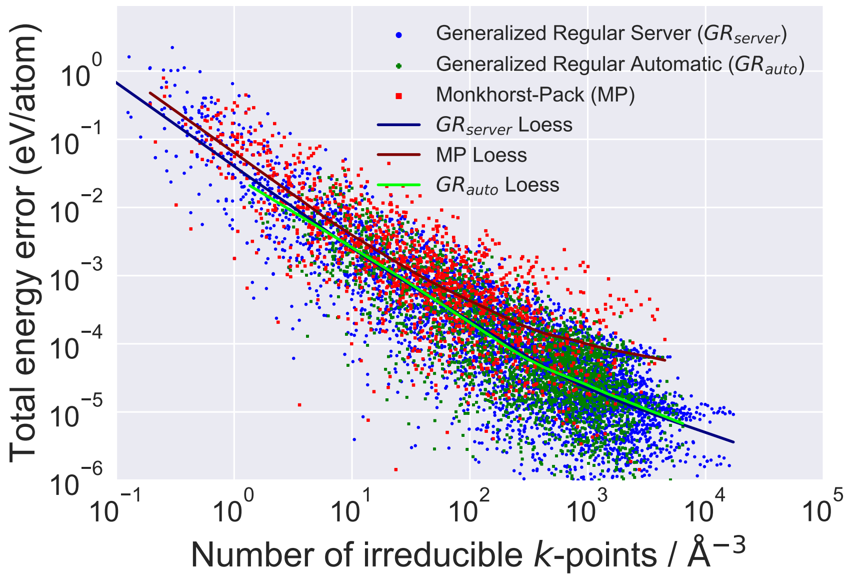


Figure 5.9 The total energy convergence with respect to irreducible **k**-point density for MP, GR_{auto} and GR_{server} grids with loess regression applied.

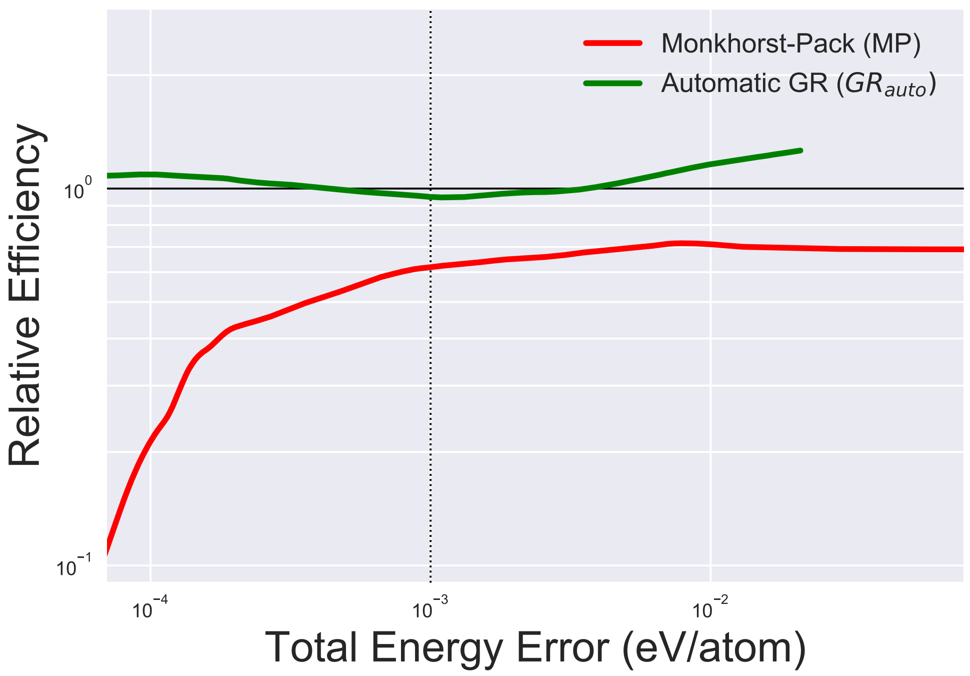


Figure 5.10 Along the y-axis are the ratios of the MP and GR_{auto} efficiencies compared to the GR_{server} grid efficiency (black horizontal line at 10^0). Total energy error (per atom) is plotted along the x-axis and decreases to the left. MP grids are $\sim$60% less efficient than both GR_{auto} and GR_{server} grids at a target accuracy of 1meV/atom. The GR_{auto} grids closely agree with GR_{auto} grids.

CHAPTER 6

Conclusion

In computational materials science Density Functional Theory (DFT) calculations are performed to determine a material's properties. The main drawback of DFT is that it can only determine the properties of a single candidate material at a time. The search space for materials is infinite; however, for any given material the number of possible atomic arrangements can be reduced to a reasonable number by selecting a lattice and the number of atoms to include in the simulations. For systems that have large amounts of configurational freedom, such as high-entropy alloys, the list of possible atomic arrangements can be too large to store in a computer. This list can be reduced by eliminating structures that are the same when a translation or rotation of the lattice is applied (these operations will change the material's properties). Enumerating the symmetrically unique configurations of atoms is a common first step in many computational studies.

When enumerating the symmetrically distinct atomic arrangements it is helpful to know how many arrangements exist. The Pòlya enumeration algorithm, found in Chapter 2, can determine this number. The algorithm allows researchers to determine if a system has too many unique arrangements before performing computations. The algorithm also allows alloy enumeration algorithms to 1) verify that they found the correct number of unique arrangements and 2) check that the list of unique arrangements will fit in memory before finding them all.

The standard approach to the alloy enumeration problem is to build a list of possible arrangements and then remove the duplicates. For systems with large amounts of configurational freedom this initial list is often too large to fit in memory even though the list of unique arrangements can be much smaller. The algorithm described in Chapter 3 overcomes this problem by using a tree search algorithm that constructs only the unique arrangements while skipping the duplicates completely. This more efficient approach also allows for new types of configurational freedom to be included in the enumeration, such as displacement directions.

The creation of the Pòlya enumeration algorithm and the new alloy enumeration algorithm allows researchers to explore more complex and larger systems than have previously been accessible to materials scientists. Both of these codes have been implemented in the `enumlib` package.

To determine material properties DFT codes perform a numerical integral of the electronic energy. The accuracy of this integral, and the resulting material properties, is directly related to the number of sample points, **k**-points, used. While this is well known in the community there have never been any detailed studies of how **k**-point sampling affects the calculations accuracy prior to this work. Chapter 4 contains a study that shows how **k**-point sampling impacts the accuracy of DFT calculations. The study also shows that the type of **k**-point grid used can impact the calculations efficiency by up to ∼60%. The most efficient grids, called *general regular* (GR) grids, are those that have the fewest symmetrically distinct **k**-points.

GR grids have not been widely adopted by the DFT community because they are difficult to generate, requiring a search over a large number of candidate grids. The algorithms described in Chapter 5 overcome this problem by only searching over **k**-point grids that have the same symmetry as the system being studied. The search over this much smaller number of candidate grids typically only takes a few seconds and generates grids that have the same efficiency as those used in the study found in Chapter 4.

In conclusion this dissertation details algorithms that use symmetry 1) to reduce the search space of materials in an efficient manner and 2) to find the optimal **k**-point grid for use in DFT calculations. These codes will allow researchers to study systems with large amounts of configurational freedom in greater detail than previously possible while optimizing the performance of the DFT calculations being performed.

APPENDIX A

Proof of Brillouin Zone Location

A.1 FORMAL PROOF

Given a point x in the space, the term *cousin* will be used for a point x' which differs from x by an element of the lattice—i.e., a coset representative or a lattice-translation equivalent point.

Let R be a basis. Let U_R denote the union of 2^d basis cells around the origin—the set of points which are expressible in terms of the basis R with all coefficients having absolute value < 1. Let V denote the Voronoi cell (Brillouin zone)—the set of all points in the space which are closer to the origin than any other lattice point. Note that U_R depends on the basis R, but V depends only on the lattice. Note also that both U_R and V are convex sets.

The claim (in two and three dimensions) is that if R is a Minkowski basis, then $V \subseteq U_R$. Arguing by contrapositive— if $V \not\subseteq U_R$, then the basis is not Minkowski reduced.

If $V \not\subseteq U_R$ then V must intersect the boundary of U_R, so there exist points on the boundary of U_R which are closer to the origin than to any other lattice points. Equivalently, those points are closer to the origin than any of their cousins.

Note that among the cousins of any point on the boundary of U_R, there is always a closest to the origin. But usually points on the boundary will have closer cousins in the interior. But if $V \not\subseteq U_R$ there must be points on the boundary which have no closer cousins in the interior of U_R. In other words, there are points (at least one) on the boundary such that *all* of its cousins in the interior of U_R are farther from the origin.

A.2 2D ARGUMENT

Let $\vec{r}_1$ and $\vec{r}_2$ be basis elements of R. Assuming that $V \not\subseteq U_R$ there must be a point x on the boundary of U_R whose cousins are all farther from the origin than x.

Without loss of generality (re-label the basis if necessary), we may express one of the bounding edges of U_R as $x = \vec{r}_1 + \lambda\vec{r}_2$ where $\lambda \in [-1, 1]$. One of its interior cousins is $x' = \lambda\vec{r}_2$, which is illustrated in Fig. A.1, then (since x' must be farther from the origin)

$$\begin{aligned}
x^2 &< x'^2 \\
(\vec{r}_1 + \lambda\vec{r}_2)^2 &< (\lambda\vec{r}_2)^2 \\
\vec{r}_1^2 + 2\lambda\vec{r}_1\cdot\vec{r}_2 + \lambda^2\vec{r}_2^2 &< \lambda^2\vec{r}_2^2 \\
\vec{r}_1^2 &< -2\lambda\vec{r}_1\cdot\vec{r}_2
\end{aligned}$$

Since the expression on the left-hand side is greater than zero, the expression on the right-hand side must be also and taking the absolute value of both sides does not change the inequality:

$$|\vec{r}_1^2| < |-2\lambda\vec{r}_1\cdot\vec{r}_2| \implies |\vec{r}_1|^2 < 2|\lambda||\vec{r}_1\cdot\vec{r}_2|.$$

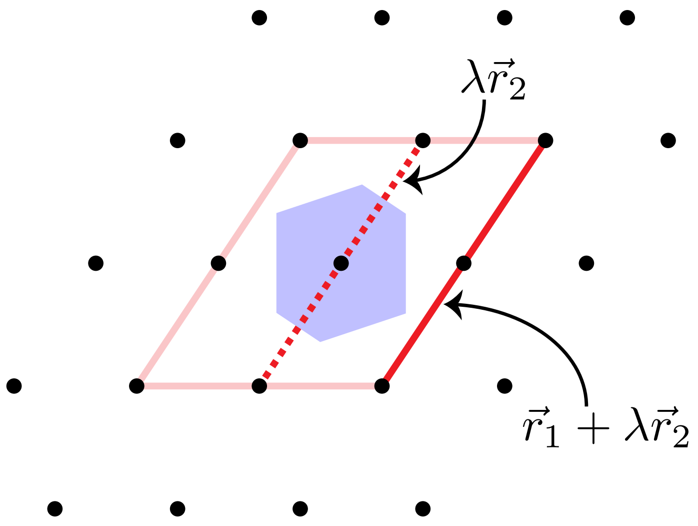


Figure A.1 Each point along the boundary of U_R has at least one interior cousin closer to the origin when R is Minkowski reduced. For the points along the edge in red, these interior cousins are the points along the dashed red line.

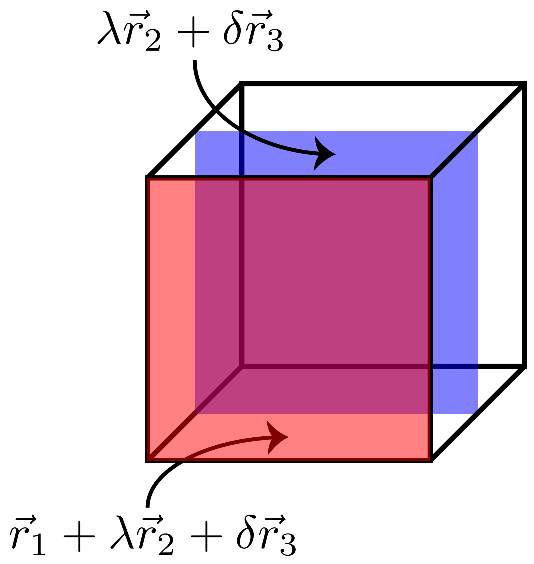

$\lambda\vec{r}_2 + \delta\vec{r}_3$
$\vec{r}_1 + \lambda\vec{r}_2 + \delta\vec{r}_3$

Figure A.2 Each point along the boundary of U_R, the edges of which are shown in black, has at least one interior cousin closer to the origin when R is Minkowski reduced. For the points on the bounding plane in red, the interior cousins are the points on the plane in blue. (The origin is contained in the blue plane.)

Considering the worst case scenario of $\lambda = 1$ gives

$$\frac{|\vec{r}_1|}{2} < \frac{|\vec{r}_1 \cdot \vec{r}_2|}{|\vec{r}_1|}, \tag{A.1}$$

which violates the condition of a Minkowski basis $|\vec{r}_1 \cdot \vec{r}_2|/|\vec{r}_1| < |\vec{r}_1|/2$. The remaining three boundaries are similar to the one just considered, the only differences being permutations of the basis elements $\vec{r}_1$ and $\vec{r}_2$ and possibly changes of sign. When applying the same reasoning to the other edges we arrive at the same contradiction. Hence, the points on the boundary of U_R are closer to the origin than interior cousins, $V \nsubseteq U_R$, only when the basis R is not Minkowski reduced. If R is Minkowski reduced, all points on the boundary of U_R have interior cousins that lie closer to the origin and $V \subseteq U_R$.

A.3 3D ARGUMENT

Let $\vec{r}_1, \vec{r}_2$, and $\vec{r}_3$ be the basis elements of R, and suppose (relabeling the basis vectors if necessary) that $x = \vec{r}_1 + \lambda\vec{r}_2 + \delta\vec{r}_3$ (where λ and δ are elements of $[-1,1]$) is a point on the boundary of U_R which is closer to the origin than its interior cousins.

One of those cousins is a plane through the origin $x' = \lambda\vec{r}_2 + \delta\vec{r}_3$. The boundary and cousin planes are shown in Fig. A.2. Thus

$$x^2 < x'^2$$
$$(\vec{r}_1 + \lambda\vec{r}_2 + \delta\vec{r}_3)^2 < (\lambda\vec{r}_2 + \delta\vec{r}_3)^2.$$

Simplifying this expression gives

$$\vec{r}_1^2 < -2\lambda\vec{r}_1 \cdot \vec{r}_2 - 2\delta\vec{r}_1 \cdot \tag{A.2}$$

Since the expression on the left-hand side is greater than zero, the expression on the right-hand side must be also and taking the absolute value of both sides does not change the inequality:

$$|\vec{r}_1^2| < |-2\lambda\vec{r}_1 \cdot \vec{r}_2 - 2\delta\vec{r}_1 \cdot \vec{r}_3| \implies |\vec{r}_1|^2 < |2\lambda\vec{r}_1 \cdot \vec{r}_2 + 2\delta\vec{r}_1 \cdot \vec{r}_3| \tag{A.3}$$

Using the triangle inequality to simplify the right hand side of Eq. A.3 makes it more likely that the inequality is satisfied:

$$|\vec{r}_1|^2 < |2\lambda\vec{r}_1 \cdot \vec{r}_2 + 2\delta\vec{r}_1 \cdot \vec{r}_3|$$
$$|\vec{r}_1|^2 < 2|\lambda||\vec{r}_1 \cdot \vec{r}_2| + 2|\delta||\vec{r}_1 \cdot \vec{r}_3| \tag{A.4}$$

Since the expression in Eq. A.4 does not depend on the sign of λ or δ, λ and δ can be restricted to positive values within $[0,1]$ without loss of generality. Consider now *another* cousin that lies within U_R and on the same plane as x': $x'' = (\lambda-1)\vec{r}_2 + (\delta-1)\vec{r}_3$. Repeating the same process for x' with x'' gives

$$|\vec{r}_1|^2 < 2|\lambda-1||\vec{r}_1 \cdot \vec{r}_2| + 2|\delta-1||\vec{r}_1 \cdot \vec{r}_3| \tag{A.5}$$

Combining Eqs. A.4 and A.5 gives

$$\begin{aligned}
|\vec{r}_1|^2 &< (|\lambda|+|\lambda-1|)|\vec{r}_1\cdot\vec{r}_2|+(|\delta|+|\delta-1|)|\vec{r}_1\cdot\vec{r}_3| \\
|\vec{r}_1| &< \frac{|\vec{r}_1\cdot\vec{r}_2|}{|\vec{r}_1|}+\frac{|\vec{r}_1\cdot\vec{r}_3|}{|\vec{r}_1|} \qquad \text{(A.6)}
\end{aligned}$$

Assuming $\{\vec{r}_1,\vec{r}_2,\vec{r}_3\}$ forms a Minkowski basis, and plugging in the largest possible values for all quantities under this assumption on the right-hand side of Eq. A.6 gives the contradiction $|\vec{r}_1| < |\vec{r}_1|$. The remaining seven bounding planes are similar to the one just considered, the only differences being permutations of the basis elements $\vec{r}_1$, $\vec{r}_2$, and $\vec{r}_3$ and changes of sign. The same contradiction arises when applying the same reasoning to the other planes. Hence, the points on the boundary of U_R are closer to the origin than interior cousins, $V \nsubseteq U_R$, only when the basis R is not Minkowski reduced. If R is Minkowski reduced, all points on the boundary of U_R have interior cousins that lie closer to the origin and $V \subseteq U_R$.

APPENDIX B

Groups, Matrices, and Lattices in Smith Normal Form.

The discussion below is limited to three-dimensions though the arguments easily generalize to higher dimensions. The purpose of the discussion below is to help the reader make the connection between groups and integer matrices. The Smith Normal Form is a key concept to make this connection.

This discussion shows that an association can be made between a single, finite group and the lattice sites within one tile (i.e., one unit cell) of a superlattice. For application, this tile is the unit cell of the grid generating vectors and the superlattice is the reciprocal cell. The association between the group and the lattice sites is a homomorphism that maps each lattice site to an element of the group. If two points are translationally equivalent (same site but in two different tiles) they will map to the same element of the group. This homomorphism is the key ingredient to constructing the hash function (see Eq. 5.4) that enables a perfect hash table where points are listed consecutively, from 1 to N. The following details how to find the homomorphism between the group and the lattice sites.

B.1 GROUPS IN SMITH NORMAL FORM

Begin with the simplest case. Let $\mathbb{N}$ be a non-singular 3×3 matrix of integers. Its columns represent the basis for a subgroup $\mathscr{L}_N$ of the group $\mathbf{Z}^3$ (where $\mathbf{Z}$ is the set of all integers, and the group operation is addition). The two lattices whose symmetries are represented by these two groups are the "simple cubic" lattice of all points with all integer coordinates and its *superlattice*[1] whose basis is given by the columns of $\mathbb{N}$. Since $\mathbf{Z}^3$ and its subgroups are Abelian, all the subgroups are *normal* so there exists a quotient group $G=\mathbf{Z}^3/\mathscr{L}_N$, and that group is *finite*.

Note that the *cosets* which form the elements of that quotient group are simply the distinct translates of the lattice $\mathscr{L}_N$ within $\mathbf{Z}^3$. In fact, each coset has exactly one representative in each unit cell, so the order of G is equal to the volume of a unit cell (the absolute value of the determinant of $\mathbb{N}$). Since the quotient group G is finite, and Abelian, it must be a direct sum of cyclic groups (by the Fundamental Theorem of Finite Abelian Groups).

One canonical form for direct sums of groups is called Smith Normal Form, where the direct summands are ordered so that each summand divides the next. In other words, $G\cong\mathbf{Z}_{m_1}\oplus\mathbf{Z}_{m_2}\oplus\cdots\oplus\mathbf{Z}_{m_k}$ where $m_1|m_2|\ldots m_{k-1}|m_k$ and (of course) $\prod m_i=|G|$. Any finite Abelian group can be uniquely written in this form. (Isomorphic groups will yield the same "invariant factors" m_1, m_2,…,m_k when written in this form.)

Note that, since $G=\mathbf{Z}^3/\mathscr{L}_N$, there must be a homomorphism from $\mathbf{Z}^3$ onto G, having $\mathscr{L}_N$ as its kernel. In other words, $\mathscr{L}_N=\{p\in\mathbf{Z}^3\ :\ \psi(p)=0\}$. Our task is to find the direct-sum

[1]In the mathematical literature, and in some of the crystallography literature, these "superlattices" are referred to as sublattices. The group associated with a "superlattice" is a *subgroup* of the group associated with the parent lattice. Although this nomenclature (subgroups, sublattices) is more correct from a mathematical or group theory point of view, the nomenclature typically seen in the physics literature is used where a lattice or a structure whose volume is larger than that of the parent is referred as a superlattice.

representation of the quotient group $\mathbf{Z}^3/\mathscr{L}_N$, and also to find the homomorphism ψ which maps the points of $\mathbf{Z}^3$ onto the group (in such a way that $\psi(p) = 0$ iff $p \in \mathscr{L}_N$). The elements of the group can then be worked with as proxies for the **k**-points inside the reciprocal cell.

B.2 MATRICES IN SMITH NORMAL FORM

There is a useful connection between the SNF for Abelian groups and the SNF for integer matrices. As the reader may infer, the SNF form of the basis matrix $\mathbb{N}$ effectively tells one how to represent the quotient group $\mathbf{Z}^3/\mathscr{L}_N$ as a direct sum of cyclic groups in Smith Normal Form, and, as shown in the following section, the row operations used to create the SNF of $\mathbb{N}$ give the homomorphism ψ suggested above.

The Connection Between SNF Groups and SNF Matrices

In the matrix case, since the operations are elementary row and column operations, $\mathbb{D} = \mathbb{A}\mathbb{N}\mathbb{B}$ where $\mathbb{A}$ and $\mathbb{B}$ are integer matrices with determinant ± 1 representing the accumulated row operations and column operations respectively. The matrix $\mathbb{D}$ is completely determined by $\mathbb{N}$, but the matrices $\mathbb{A}$ and $\mathbb{B}$ depend on the algorithm used to arrive at the Smith Normal Form of $\mathbb{N}$. A different implementation might yield $\mathbb{D} = \mathbb{A}'\mathbb{N}\mathbb{B}'$ (same $\mathbb{N}$ and same $\mathbb{D}$, but different $\mathbb{A}$ and $\mathbb{B}$).

Note that, since $\mathbb{B}$ represents elementary column operations, the product $\mathbb{N}\mathbb{B}$ simply represents a change of basis from $\mathbb{N}$ to a new basis $\mathbb{N}' = \mathbb{N}\mathbb{B}$. In other words, the columns of $\mathbb{N}'$ are still a basis for $\mathscr{L}_N$. But the new basis has the property that $\mathbb{A}\mathbb{N}' = \mathbb{D}$. That means that every element $\vec{w} = \mathbb{N}'\vec{z}$ of $\mathscr{L}_N$ (where $\vec{z}$ is some element of $\mathbb{Z}^3$) will satisfy the equation $\mathbb{A}\vec{w} = \mathbb{D}\vec{z} = \begin{pmatrix} \mathbb{D}_{11}z_1 \\ \mathbb{D}_{22}z_2 \\ \mathbb{D}_{33}z_3 \end{pmatrix}$.
In other words, $\mathbb{A}\vec{w}$ will be a vector whose entries are *multiples* of the corresponding diagonal entries in $\mathbb{D}$.

To put it another way, define $*$ to be the operation that maps

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{B.1}$$

in $\mathbb{Z}^3$ to

$$\vec{x}^* = \begin{pmatrix} x_1 \pmod{\mathbb{D}_{11}} \\ x_2 \pmod{\mathbb{D}_{22}} \\ x_3 \pmod{\mathbb{D}_{33}} \end{pmatrix}^T \tag{B.2}$$

Then it is clear that $\vec{w} \in \mathscr{L}_N$ iff $(\mathbb{A}w)^* = (0,0,0)$ (the zero-element in the group $G_0 = \mathbb{Z}_{\mathbb{D}_{11}} \oplus \mathbb{Z}_{\mathbb{D}_{22}} \oplus \mathbb{Z}_{\mathbb{D}_{33}}$).

That suggests letting $\psi(\vec{w}) = (\mathbb{A}\vec{w})^*$, a homomorphism from $\mathbf{Z}^3$ onto the direct-sum G_0. Then, since that homomorphism is easily shown to be onto, and its kernel is $\mathscr{L}_N$, it can be seen (by the First Isomorphism Theorem of group theory) that $G_0 \cong \mathbf{Z}^3/\mathscr{L}_N$, and ψ is precisely the desired homomorphism.

Thus the two versions of SNF have been connected. The matrix algorithm provides the SNF description of the quotient group by the diagonal entries in $\mathbb{D}$, and the transition matrix $\mathbb{A}$ provides the homomorphism which maps the parent lattice onto the group.

An example. Let $\mathbb{N} = \begin{pmatrix} 1 & 2 & -1 \\ 1 & 4 & -3 \\ 0 & 2 & 4 \end{pmatrix}$. This describes a lattice $\mathscr{L}_N$ which contains the points $\vec{p}_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$, $\vec{p}_2 = \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix}$, and $\vec{p}_3 = \begin{pmatrix} -1 \\ -3 \\ 4 \end{pmatrix}$, and *all* the points which are integer linear combinations of those three points. The matrix $\mathbb{N}$ has determinant 12, which must be the volume of each lattice tile—and it is also the order of the quotient group $\mathbf{Z}^3/\mathscr{L}_N$.

Using the SNF algorithm to diagonalize this basis matrix yields $\mathbb{D} = \mathbb{A}\mathbb{N}\mathbb{B}$ where $\mathbb{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{pmatrix}$, with $\mathbb{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -1 & -2 \end{pmatrix}$ and $\mathbb{B} = \begin{pmatrix} 1 & 7 & 11 \\ 0 & -1 & -2 \\ 0 & 1 & 1 \end{pmatrix}$.

Thus the quotient group is $G = \mathbf{Z}^3/\mathscr{L}_N \cong \mathbf{Z}_1 \oplus \mathbf{Z}_2 \oplus \mathbf{Z}_6 \cong \mathbf{Z}_2 \oplus \mathbf{Z}_6$.

Further, from the matrix $\mathbb{A}$, the homomorphism projecting $\mathbf{Z}^3$ onto the quotient group can be obtained, with kernel $\mathscr{L}_N$. If $\vec{w} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ then $\mathbb{A}\vec{w} = \begin{pmatrix} y \\ z \\ x-y-2z \end{pmatrix}$ and thus

$$\begin{aligned} \psi(\vec{w}) &= (\mathbb{A}\vec{w})^* \\ &= \begin{pmatrix} y \pmod 1 \\ z \pmod 2 \\ x-y-2z \pmod 6 \end{pmatrix}^T \\ &= \big(z \pmod 2, x+5y+4z \pmod 6\big) \end{aligned}$$

(noting that anything mod 1 is zero).

Note that this homomorphism provides a different, but convenient, way to describe the superlattice. Since $\mathscr{L}_N$ is the kernel of ψ, it is comprised of the points $(x,y,z) \in \mathbf{Z}^3$ which satisfy the simultaneous congruences $z \equiv 0$ (mod 2) and $x+5y+4z \equiv 0$ (mod 6). Noting that all three basis points p_1, p_2 and p_3 satisfy these congruences, and thus so will all their integer linear combinations (all points in $\mathscr{L}_N$).

Algorithmic variation. In the example computed above, a different application of the SNF matrix algorithm, with the same $\mathbb{N}$, might have yielded the same diagonal matrix $\mathbb{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{pmatrix}$, but different $\mathbb{A} = \begin{pmatrix} 1 & 0 & 0 \\ -5 & 3 & 1 \\ -2 & 2 & 1 \end{pmatrix}$ and $\mathbb{B} = \begin{pmatrix} 0 & -1 & 2 \\ 0 & 0 & 1 \\ -1 & -1 & 4 \end{pmatrix}$, which would change the homomorphism to $(x,y,z) \mapsto \big(-5x+3y+z \pmod 2,\ -2x+2y+z \pmod 6\big) = \big(x+y+z \pmod 2,\ 4x+$

$2y+z \pmod 6\big)$.The new homomorphism is different, since $(1,0,1) \mapsto (0,5)$ now, where previously $(1,2,3) \mapsto (1,5)$ (for example), but the *kernel* is the same. In fact the two homomorphisms are related via an automorphism of the group G.

APPENDIX C

Integer Relations for all Niggli Cells

The following are the Niggli basis and the resulting integer relationships used in the final implementation of the GR on the fly algorithm. The relationships are grouped by crystal class and Niggli case number. Each subsection starts with the associated Niggli case numbers, followed by the chosen basis, then the resulting integer relations from 5.10. Some sections have multiple Niggli cases that have the same representations of the symmetry group $\mathbb{X}$s even though they have different lattice basis. For those cases multiple bases are provided, one for each Niggli case that has the equivalent representation. Please note that there is nothing special about the atomic basis displayed here, they are simply basis that produced a convenient form of the symmetry group $\mathbb{X}$s.

C.1 SIMPLE CUBIC

Niggli Case 3

$$\mathbb{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{C.1}$$

$$\begin{aligned}
\alpha_1 &= \left(0,\ \tfrac{d}{a}\right) \\
\alpha_2 &= \left(0,\ \tfrac{e}{a}\right) \\
\alpha_3 &= \left(0,\ \tfrac{f}{a}\right) \\
\beta_1 &= \left(\tfrac{b-d}{c},\ \tfrac{-a-b\alpha_1}{c}\right) \\
\beta_2 &= \left(\tfrac{-e}{c},\ \tfrac{-b\alpha_2}{c}\right) \\
\beta_3 &= \left(\tfrac{-f}{c},\ \tfrac{-b\alpha_3}{c}\right) \\
\gamma_1 &= \left(\tfrac{-b+d-e\beta_1}{f},\ \tfrac{b-d\alpha_2-e\beta_2}{f}\right) \\
\gamma_2 &= \left(\tfrac{-c-e\beta_1}{f},\ \tfrac{c-d\alpha_2-e\beta_2}{f}\right)
\end{aligned} \tag{C.2}$$

C.2 BODY CENTERED CUBIC

Niggli Case 5

$$\mathbb{A} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} \tag{C.3}$$

$$\begin{aligned}
\alpha_1 &= \left(\tfrac{-d}{a},\, 0\right)\\
\alpha_2 &= \left(\tfrac{e}{a},\, 0\right)\\
\alpha_3 &= \left(\tfrac{f}{a},\, 0\right)\\
\beta_1 &= \left(\tfrac{-a+b+b\alpha_1}{c},\, \tfrac{b-d}{c}\right)\\
\beta_2 &= \left(\tfrac{-b\alpha_2}{c},\, \tfrac{-e}{c}\right)\\
\beta_3 &= \left(\tfrac{-bf}{ac},\, \tfrac{-f}{c}\right)\\
\gamma_1 &= \left(\tfrac{b+d-d\alpha_1-e\beta_1}{f},\, \tfrac{-a-b+d-e\beta_1}{f}\right)\\
\gamma_2 &= \left(\tfrac{-c-d\alpha_1-e\beta_2}{f},\, \tfrac{-e-e\beta_2}{f}\right)
\end{aligned} \tag{C.4}$$

C.3 FACE CENTERED CUBIC

Niggli Case 1

$$\mathbb{A} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \tag{C.5}$$

$$\begin{aligned}
\alpha_1 &= \left(\tfrac{-b-d}{a},\, 0\right)\\
\alpha_2 &= \left(\tfrac{-c-e}{a},\, 0\right)\\
\alpha_3 &= \left(\tfrac{-f}{a},\, 0\right)\\
\beta_1 &= \left(\tfrac{2b-b\alpha_1}{c},\, \tfrac{b-d}{c}\right)\\
\beta_2 &= \left(\tfrac{-b\alpha_2}{c},\, \tfrac{-e}{c}\right)\\
\beta_3 &= \left(\tfrac{bf}{ac},\, \tfrac{-f}{c}\right)\\
\gamma_1 &= \left(\tfrac{2d-d\alpha_1-e\beta_1}{f},\, \tfrac{a+b+2d-e\beta_2}{f}\right)\\
\gamma_2 &= \left(\tfrac{-d\alpha_2+e\beta_2}{f},\, \tfrac{c+e+\frac{e^2}{c}}{f}\right)
\end{aligned} \tag{C.6}$$

C.4 HEXAGONAL

Niggli Case 12

$$\mathbb{A} = \begin{pmatrix} 1 & \frac{1}{2} & 0 \\ 0 & -\frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 2 \end{pmatrix} \tag{C.7}$$

$$
\begin{aligned}
\alpha_1 &= \left(\tfrac{-b}{a},\ \tfrac{-b}{a},\ 0\right)\\
\alpha_2 &= \left(\tfrac{-c}{a},\ \tfrac{-c}{a},\ 0\right)\\
\alpha_3 &= \beta_3 = \left(0,\ 0,\ 0\right)\\
\beta_1 &= \left(\tfrac{2b-b\alpha_1}{c},\ \tfrac{2b-b\alpha_1}{c},\ \tfrac{a+2b}{c}\right)\\
\beta_2 &= \left(-\alpha_1,\ -\alpha_1,\ 0\right)\\
\gamma_1 &= \left(\tfrac{d\beta_2-e\beta_1}{f},\ \tfrac{2d+d\beta_2-e\beta_1}{f},\ \tfrac{-e\beta_1}{f}\right)\\
\gamma_2 &= \left(\tfrac{-d\alpha_2-2e-e\beta_2}{f},\ \tfrac{-d\alpha_2-e\beta_2}{f},\ \tfrac{-2e}{f}\right)
\end{aligned}
\tag{C.8}
$$

Niggli Case 22

$$
\mathbb{A} = \begin{pmatrix} 0 & 1 & -\frac{1}{2} \\ 0 & 0 & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & 0 & 0 \end{pmatrix}
\tag{C.9}
$$

$$
\begin{aligned}
\alpha_1 &= \alpha_2 = \alpha_3 = \left(0,\ 0,\ 0\right)\\
\beta_1 &= \left(0,\ \tfrac{d}{c},\ 0\right)\\
\beta_2 &= \left(0,\ \tfrac{e}{c},\ 0\right)\\
\beta_3 &= \left(0,\ \tfrac{f}{c},\ 0\right)\\
\gamma_1 &= \left(\tfrac{-b+2d}{f},\ \tfrac{-b+d-d\beta_2}{f},\ 0\right)\\
\gamma_2 &= \left(\tfrac{-c+2e}{f},\ \tfrac{-c+e-e\beta_2}{f},\ 0\right)
\end{aligned}
\tag{C.10}
$$

C.5 RHOMBOHEDRAL

Niggli Cases 2 and 4

$$
\mathbb{A}_2 = \begin{pmatrix} -1.11652 & 0 & 1 \\ -0.610985 & -1.32288 & 1.32288 \\ 0.616515 & -\frac{1}{2} & \frac{3}{2} \end{pmatrix}
\tag{C.11}
$$

$$
\mathbb{A}_4 = \begin{pmatrix} -0.548584 & 0 & 1 \\ 0.774292 & -1.32288 & 1.32288 \\ 1.04858 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}
\tag{C.12}
$$

$$\begin{aligned}
\alpha_1 &= \left(0,\ \tfrac{-d}{a}\right)\\
\alpha_2 &= \left(0,\ \tfrac{-e}{a}\right)\\
\alpha_3 &= \left(0,\ \tfrac{-f}{a}\right)\\
\beta_1 &= \left(\tfrac{2b}{c},\ \tfrac{-a+b-d-b\alpha_1}{c}\right)\\
\beta_2 &= \left(0,\ \tfrac{-e-b\alpha_2}{c}\right)\\
\beta_3 &= \left(0,\ \tfrac{-f-b\alpha_3}{c}\right)\\
\gamma_1 &= \left(\tfrac{b-e\beta_1}{f},\ \tfrac{-a-d\alpha_1-e\beta_1}{f}\right)\\
\gamma_2 &= \left(\tfrac{c-2e}{f},\ \tfrac{-e-d\alpha_2-e\beta_2}{f}\right)
\end{aligned} \tag{C.13}$$

Niggli Cases 9 and 24

$$\mathbb{A}_9 = \begin{pmatrix} 1 & 2 & 4\\ 2 & 1 & 3\\ 2 & 2 & 3 \end{pmatrix} \tag{C.14}$$

$$\mathbb{A}_{24} = \begin{pmatrix} -0.255922 & 1.51184 & 1.255922\\ -1.44338 & 0 & 1.44338\\ 0.92259 & -0.845178 & 0.07741 \end{pmatrix} \tag{C.15}$$

$$\begin{aligned}
\alpha_1 &= \left(0,\ \tfrac{-d}{a}\right)\\
\alpha_2 &= \left(0,\ \tfrac{-e}{a}\right)\\
\alpha_3 &= \left(0,\ \tfrac{-f}{a}\right)\\
\beta_1 &= \left(\tfrac{2b}{c},\ \tfrac{-a+b+d-b\alpha_1}{c}\right)\\
\beta_2 &= \left(0,\ \tfrac{e-b\alpha_2}{c}\right)\\
\beta_3 &= \left(0,\ \tfrac{f-b\alpha_3}{c}\right)\\
\gamma_1 &= \left(\tfrac{-b-e\beta_1}{f},\ \tfrac{-b-d-d\alpha_1-e\beta_1}{f}\right)\\
\gamma_2 &= \left(\tfrac{-c-2e}{f},\ \tfrac{-c-2e-d\alpha_2-e\beta_2}{f}\right)
\end{aligned} \tag{C.16}$$

C.6 SIMPLE TETRAGONAL

Niggli Case 11

$$\mathbb{A} = \begin{pmatrix} 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 2 \end{pmatrix} \tag{C.17}$$

$$\begin{aligned}
\alpha_1 &= \left(0,\ 0,\ \tfrac{-b}{a}\right)\\
\alpha_2 &= \left(0,\ 0, \tfrac{-c}{a}\right)\\
\alpha_3 = \beta_3 &= \left(0,\ 0,\ 0\right)\\
\beta_1 &= \left(0,\ \tfrac{2b}{c},\ \tfrac{-e-b\alpha_1}{c}\right)\\
\beta_2 &= \left(0,\ 0,\ -\alpha_1\right)\\
\gamma_1 &= \left(0,\ \tfrac{-e\beta_1}{f},\ \tfrac{-d-d\alpha_1-e\beta_1}{f}\right)\\
\gamma_2 &= \left(0,\ \tfrac{-2e}{f},\ \tfrac{-d\alpha_2-e+-e\beta_2}{f}\right)
\end{aligned} \tag{C.18}$$

Niggli Case 21

$$\mathbb{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{1}{2} & 0 & 0 \end{pmatrix} \tag{C.19}$$

$$\begin{aligned}
\alpha_1 = \alpha_2 = \alpha_3 &= \left(0,\ 0,\ 0\right)\\
\beta_1 &= \left(0,\ \tfrac{2b}{c},\ \tfrac{b-d}{c}\right)\\
\beta_2 &= \left(0,\ 0,\ \tfrac{-e}{c}\right)\\
\beta_2 &= \left(0,\ 0,\ \tfrac{-f}{c}\right)\\
\gamma_1 &= \left(0,\ \tfrac{2d-e\beta_1}{f},\ \tfrac{-b+d-e\beta_1}{f}\right)\\
\gamma_2 &= \left(0,\ 0,\ \tfrac{-c-e\beta_2}{f}\right)
\end{aligned} \tag{C.20}$$

C.7 BODY CENTERED TETRAGONAL

Niggli Cases 6, 7, 15, and 18

$$\mathbb{A}_6 = \begin{pmatrix} 1.80278 & 2.80278 & 0.80278 \\ -1.47253 & 0.13535 & -0.47253 \\ 0.762655 & -0.791285 & 2.762655 \end{pmatrix} \tag{C.21}$$

$$\mathbb{A}_7 = \begin{pmatrix} 1.95095 & 0 & 0.95095 \\ 1.19163 & 2.60788 & -0.41625 \\ 0.879663 & 0.44606 & 2.433603 \end{pmatrix} \tag{C.22}$$

$$\mathbb{A}_{15} = \begin{pmatrix} -1 & 2 & -2 \\ -1 & -2 & 0 \\ 2 & 0 & 0 \end{pmatrix} \tag{C.23}$$

$$\mathbb{A}_{18} = \begin{pmatrix} -2 & -3 & -1 \\ -1 & 1 & -3 \\ 1 & 0 & 0 \end{pmatrix} \tag{C.24}$$

$$\begin{aligned}
\alpha_1 &= \alpha_2 = \alpha_3 = \left(0,\ 0,\ 0\right) \\
\beta_1 &= \left(0,\ \tfrac{b-d}{c},\ 0\right) \\
\beta_2 &= \left(0,\ \tfrac{-e}{c},\ 0\right) \\
\beta_3 &= \left(0,\ \tfrac{-f}{c},\ 0\right) \\
\gamma_1 &= \left(0,\ \tfrac{-b+d-e\beta_1}{f},\ \tfrac{a+2d}{f}\right) \\
\gamma_2 &= \left(0,\ \tfrac{-c-e\beta_2}{f},\ \tfrac{2e}{f}\right)
\end{aligned} \tag{C.25}$$

C.8 SIMPLE ORTHORHOMBIC

Niggli Case 32

$$\mathbb{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \tag{C.26}$$

$$\begin{aligned}
\alpha_1 &= \alpha_2 = \alpha_3 = \beta_2 = \beta_3 = \left(0,\ 0,\ 0\right) \\
\beta_1 &= \left(0,\ 0,\ \tfrac{2b}{c}\right) \\
\gamma_1 &= \left(0,\ \tfrac{2d}{f},\ \tfrac{-e\beta_1}{f}\right) \\
\gamma_2 &= \left(0,\ \tfrac{2e}{f},\ \tfrac{-2e}{f}\right)
\end{aligned} \tag{C.27}$$

C.9 BASE CENTERED ORTHORHOMBIC

Niggli Cases 13 and 38

$$\mathbb{A}_{13} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & -1.73205 \\ 1 & -1 & 1.73205 \end{pmatrix} \tag{C.28}$$

$$\mathbb{A}_{38} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 1 & -1 & 0 \\ 1 & 0 & 3 \end{pmatrix} \tag{C.29}$$

$$\begin{aligned}
\alpha_1 &= \left(0,\ 0,\ \tfrac{-b}{a}\right)\\
\alpha_2 &= \left(0,\ 0,\ \tfrac{-c}{a}\right)\\
\alpha_3 = \beta_3 &= \left(0,\ 0,\ 0\right)\\
\beta_1 &= \left(0,\ 0,\ \tfrac{-a-b\alpha_1}{c}\right)\\
\beta_2 &= \left(0,\ 0,\ -\alpha_1\right)\\
\gamma_1 &= \left(0,\ \tfrac{2d}{f},\ \tfrac{-b-d\alpha_1-e\beta_1}{f}\right)\\
\gamma_2 &= \left(0,\ \tfrac{2e}{f},\ \tfrac{-d\alpha_2-e-e\beta_2}{f}\right)
\end{aligned} \tag{C.30}$$

Niggli Case 23

$$\mathbb{A} = \begin{pmatrix} -\frac{1}{3} & 1 & 2 \\ -1.54116 & 1 & -1 \\ 1.87449 & 1 & 1 \end{pmatrix} \tag{C.31}$$

$$\begin{aligned}
\alpha_1 &= \left(0,\ \tfrac{d}{a},\ 0\right)\\
\alpha_2 &= \left(0,\ \tfrac{e}{a},\ 0\right)\\
\alpha_3 &= \left(0,\ \tfrac{f}{a},\ 0\right)\\
\beta_1 &= \left(0,\ \tfrac{-b\alpha_1}{c},\ \tfrac{2b}{c}\right)\\
\beta_2 &= \left(0,\ \tfrac{-b\alpha_2}{c},\ 0\right)\\
\beta_3 &= \left(0,\ \tfrac{-f}{c},\ \tfrac{f}{c}\right)\\
\gamma_1 &= \left(0,\ \tfrac{-b+d-e\beta_1}{f},\ \tfrac{b+d-e\beta_1}{f}\right)\\
\gamma_2 &= \left(0,\ \tfrac{-c-e\beta_2}{f},\ \tfrac{c-e\beta_2}{f}\right)
\end{aligned} \tag{C.32}$$

Niggli Case 36

$$\mathbb{A} = \begin{pmatrix} 1 & 1.41421 & -1.43541 \\ 1 & -1.41421 & -1.43541 \\ 1 & 0 & 1.37083 \end{pmatrix} \tag{C.33}$$

$$\begin{aligned}
\alpha_1 = \alpha_2 = \alpha_3 &= \left(0,\ 0,\ 0\right)\\
\beta_1 &= \left(0,\ \tfrac{b-d}{c},\ \tfrac{b+d}{c}\right)\\
\beta_2 &= \left(0,\ \tfrac{-e}{c},\ \tfrac{e}{c}\right)\\
\beta_3 &= \left(0,\ \tfrac{-f}{c},\ \tfrac{f}{c}\right)\\
\gamma_1 &= \left(0,\ \tfrac{2d-d\alpha_1-e\beta_1}{f},\ \tfrac{-e\beta_1}{f}\right)\\
\gamma_2 &= \left(0,\ \tfrac{2e-d\alpha_2-e\beta_2}{f},\ \tfrac{-2e}{f}\right)
\end{aligned} \tag{C.34}$$

Niggli Case 40

$$\mathbb{A} = \begin{pmatrix} 1 & 1.61803 & -1.05557 \\ 1 & -0.618034 & 1.99895 \\ 1 & -1 & -0.943376 \end{pmatrix} \tag{C.35}$$

$$\begin{aligned} \alpha_1 &= \alpha_2 = \alpha_3 = \left(0,\ 0,\ 0\right) \\ \beta_1 &= \left(0,\ \tfrac{d}{c},\ \tfrac{2b-d}{c}\right) \\ \beta_2 &= \left(0,\ \tfrac{e}{c},\ \tfrac{-e}{c}\right) \\ \beta_3 &= \left(0,\ \tfrac{f}{c},\ \tfrac{-f}{c}\right) \\ \gamma_1 &= \left(0,\ \tfrac{2d-e\beta_1}{f},\ \tfrac{-e\beta_1}{f}\right) \\ \gamma_2 &= \left(0,\ \tfrac{2e-e\beta_2}{f},\ \tfrac{-2e-e\beta_2}{f}\right) \end{aligned} \tag{C.36}$$

C.10 BODY CENTERED ORTHORHOMBIC

Niggli Case 8

$$\mathbb{A} = \begin{pmatrix} 1.41144 & -0.99868 & 3.41012 \\ 0.0885622 & 2.21232 & -1.1237578 \\ -2 & 1.268178 & -1.268178 \end{pmatrix} \tag{C.37}$$

$$\begin{aligned} \alpha_1 &= \alpha_2 = \alpha_3 = \left(0,\ 0,\ 0\right) \\ \beta_1 &= \left(\tfrac{-a+2b-2d}{c},\ 0,\ \tfrac{2b}{c}\right) \\ \beta_2 &= \left(\tfrac{-2e}{c},\ 0,\ 0\right) \\ \beta_3 &= \left(\tfrac{-2f}{c},\ 0,\ 0\right) \\ \gamma_1 &= \left(\tfrac{-e\beta_1}{f},\ \tfrac{2d}{f},\ \tfrac{a+2d-e\beta_1}{f}\right) \\ \gamma_2 &= \left(\tfrac{-2e-e\beta_2}{f},\ \tfrac{2e}{f},\ 0\right) \end{aligned} \tag{C.38}$$

Niggli Case 19

$$\mathbb{A} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 1 & 2 & 0 \\ \frac{3}{2} & 0 & 3 \end{pmatrix} \tag{C.39}$$

$$\begin{aligned} \alpha_1 &= \alpha_2 = \alpha_3 = \beta_2 = \beta_3 = \left(0,\ 0,\ 0\right) \\ \beta_1 &= \left(0,\ 0,\ \tfrac{a+2b}{c}\right) \\ \gamma_1 &= \left(0,\ \tfrac{a+2d}{f},\ \tfrac{-e\beta_1}{f}\right) \\ \gamma_2 &= \left(0,\ \tfrac{2e}{f},\ \tfrac{-2e}{f}\right) \end{aligned} \tag{C.40}$$

Niggli Case 42

$$\mathbb{A} = \begin{pmatrix} -1.53633 & 1 & 1.61803 \\ 1.36706 & 1 & -0.61803 \\ -1.33073 & 1 & -1 \end{pmatrix} \tag{C.41}$$

$$\begin{aligned} &\alpha_1 = \alpha_2 = \alpha_3 = \beta_2 = \beta_3 = \left(0,\ 0,\ 0\right) \\ &\beta_1 = \left(\tfrac{-a+2b}{c},\ \tfrac{-a+2b}{c},\ 0\right) \\ &\gamma_1 = \left(\tfrac{-a+2d-e\beta_1}{f},\ \tfrac{-e\beta_1}{f},\ \tfrac{-a+2d}{f}\right) \\ &\gamma_2 = \left(0,\ \tfrac{-2e}{f},\ \tfrac{2e}{f}\right) \end{aligned} \tag{C.42}$$

C.11 FACE CENTERED ORTHORHOMBIC

Niggli Cases 16 and 26

$$\mathbb{A}_{16} = \begin{pmatrix} 1.04442 & 0.779796 & 1.779796 \\ 1.43973 & -1.1789 & -0.1789 \\ 1.68415 & 1 & 0 \end{pmatrix} \tag{C.43}$$

$$\mathbb{A}_{26} = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 1 & 0 & 0 \\ \frac{3}{2} & \frac{3}{2} & 3 \end{pmatrix} \tag{C.44}$$

$$\begin{aligned} &\alpha_1 = \alpha_2 = \alpha_3 = \beta_2 = \beta_3 = \left(0,\ 0,\ 0\right) \\ &\beta_1 = \left(0,\ \tfrac{2b}{c},\ 0\right) \\ &\gamma_1 = \left(0,\ \tfrac{-b-e\beta_1}{f},\ \tfrac{a+b+2d}{f}\right) \\ &\gamma_2 = \left(0,\ \tfrac{-c-2e}{f},\ \tfrac{c+2e}{f}\right) \end{aligned} \tag{C.45}$$

C.12 SIMPLE MONOCLINIC

Niggli Case 33

$$\mathbb{A} = \begin{pmatrix} 2 & 0 & \frac{1}{2} \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \tag{C.46}$$

$$\begin{aligned} &\alpha_1 = \alpha_2 = \alpha_3 = \beta_2 = \beta_3 = \left(0,\ 0\right) \\ &\beta_1 = \left(0,\ \tfrac{2b}{c}\right) \\ &\gamma_1 = \left(0,\ \tfrac{-e\beta_1}{f}\right) \\ &\gamma_2 = \left(0,\ \tfrac{-2e}{f}\right) \end{aligned} \tag{C.47}$$

Niggli Cases 34 and 35

$$\mathbb{A}_{34} = \begin{pmatrix} 1 & 1.22474487 & -0.16598509 \\ 1 & -1.22474487 & -1.64308297 \\ 1 & -1 & 1.80906806 \end{pmatrix} \tag{C.48}$$

$$\mathbb{A}_{35} = \begin{pmatrix} -0.668912 & 1.61803 & 1 \\ 1.96676 & -0.618034 & 1 \\ -1.29785 & -1 & 1 \end{pmatrix} \tag{C.49}$$

$$\begin{aligned} &\alpha_1 = \alpha_2 = \alpha_3 = \beta_1 = \beta_2 = \beta_3 = \left(0,\ 0\right) \\ &\gamma_1 = \left(0,\ \tfrac{2d}{f}\right) \\ &\gamma_2 = \left(0,\ \tfrac{2e}{f}\right) \end{aligned} \tag{C.50}$$

C.13 BASE CENTERED MONOCLINIC

Niggli Cases 10, 14, 17, 27, 37, 39, and 41

$$\mathbb{A}_{10} = \begin{pmatrix} 1 & -1.46391 & 0 \\ -1 & 0 & 2 \\ 1 & 1.96391 & 0 \end{pmatrix} \tag{C.51}$$

$$\mathbb{A}_{14} = \begin{pmatrix} -1 & \frac{1}{2} & 0 \\ 1 & 0 & -2 \\ 0 & 2 & 0 \end{pmatrix} \tag{C.52}$$

$$\mathbb{A}_{17} = \begin{pmatrix} -1.05387 & -0.244302 & 1.809568 \\ -1.61088 & -2.77045 & -0.15957 \\ 1.51474 & 0.51474 & 0 \end{pmatrix} \tag{C.53}$$

$$\mathbb{A}_{27} = \begin{pmatrix} -1.464824 & -0.153209 & 1 \\ 0.464824 & 0.153209 & 1 \\ 1.907413 & -2.907413 & 0 \end{pmatrix} \tag{C.54}$$

$$\mathbb{A}_{37} = \begin{pmatrix} -1.79092 & 1 & 1 \\ -1.47209 & -1.41421 & 0 \\ 0.790922 & -1 & 1 \end{pmatrix} \tag{C.55}$$

$$\mathbb{A}_{39} = \begin{pmatrix} 0 & -1.66542 & 1 \\ -1.73205 & -0.672857 & 0 \\ -1 & 1.66542 & 1 \end{pmatrix} \tag{C.56}$$

$$\mathbb{A}_{41} = \begin{pmatrix} -1.85397 & 1 & 1 \\ -0.854143 & 0 & -1.41421 \\ 1.35397 & 1 & -1 \end{pmatrix} \tag{C.57}$$

$$\begin{aligned}
&\alpha_1 = \alpha_2 = \alpha_3 = \beta_1 = \beta_2 = \beta_3 = \big(0,\, 0\big)\\
&\gamma_1 = \left(\tfrac{-a+2d}{f},\, 0\right)\\
&\gamma_2 = \left(\tfrac{2e}{f},\, 0\right)
\end{aligned} \tag{C.58}$$

Niggli Cases 20 and 25

$$\mathbb{A}_{20} = \begin{pmatrix} 1 & 1.70119 & 0.69779 \\ 1 & -1.45119 & -1.4322505 \\ 1 & 1 & 3.23446 \end{pmatrix} \tag{C.59}$$

$$\mathbb{A}_{25} = \begin{pmatrix} 1 & 1.45119 & 0.28878 \\ 1 & -1.70119 & -3.26895 \\ 1 & -1 & 0.48018 \end{pmatrix} \tag{C.60}$$

$$\begin{aligned}
&\alpha_1 = \alpha_2 = \alpha_3 = \beta_2 = \beta_3 = \big(0,\, 0\big)\\
&\beta_1 = \big(0,\, \tfrac{2b}{c}\big)\\
&\gamma_1 = \left(0,\, \tfrac{-b-e\beta_1}{f}\right)\\
&\gamma_2 = \left(0,\, \tfrac{-c-2e}{f}\right)
\end{aligned} \tag{C.61}$$

Niggli Case 28

$$\mathbb{A} = \begin{pmatrix} -1.44896 & 1 & 0.342424 \\ 0.948958 & -1 & -1.342424 \\ -1 & 0 & -2.02006 \end{pmatrix} \tag{C.62}$$

$$\begin{aligned}
&\alpha_1 = \alpha_2 = \alpha_3 = \big(0,\, 0\big)\\
&\beta_1 = \big(\tfrac{-d}{c},\, 0\big)\\
&\beta_2 = \big(\tfrac{-e}{c},\, 0\big)\\
&\beta_3 = \left(\tfrac{-f}{c},\, 0\right)\\
&\gamma_1 = \left(\tfrac{2d-d\beta_1}{f},\, 0\right)\\
&\gamma_2 = \left(\tfrac{2e-e\beta_2}{f},\, 0\right)
\end{aligned} \tag{C.63}$$

Niggli Cases 29 and 30

$$\mathbb{A}_{29} = \begin{pmatrix} -0.666125 & 1 & 1.61803 \\ 1.16613 & 1 & -0.618034 \\ 2.04852 & 0 & 1 \end{pmatrix} \tag{C.64}$$

$$\mathbb{A}_{30} = \begin{pmatrix} 1 & 1.61803 & -0.0361373 \\ 1 & -0.618034 & 0.536137 \\ 0 & 1 & 2.38982 \end{pmatrix} \tag{C.65}$$

$$\begin{aligned} \alpha_1 &= \alpha_2 = \alpha_3 = \left(0,\, 0\right) \\ \beta_1 &= \left(\tfrac{-d}{c},\, 0\right) \\ \beta_2 &= \left(\tfrac{-e}{c},\, 0\right) \\ \beta_3 &= \left(\tfrac{-f}{c},\, 0\right) \\ \gamma_1 &= \left(\tfrac{2d-e\beta_1}{f},\, 0\right) \\ \gamma_2 &= \left(\tfrac{2e-e\beta_2}{f},\, 0\right) \end{aligned} \tag{C.66}$$

Niggli Case 43

$$\mathbb{A}_{43} = \begin{pmatrix} -0.39716 & 2.64194 & -1.39716 \\ -0.34718 & -0.14194 & -1.34718 \\ 2.49434 & 0 & 1.49434 \end{pmatrix} \tag{C.67}$$

$$\begin{aligned} \alpha_1 &= \alpha_2 = \alpha_3 = \left(0,\, 0\right) \\ \beta_1 &= \left(0,\, \tfrac{a+d}{c}\right) \\ \beta_2 &= \left(0,\, \tfrac{e}{c}\right) \\ \beta_3 &= \left(0,\, \tfrac{f}{c}\right) \\ \gamma_1 &= \left(0,\, \tfrac{2a+2d-e\beta_1}{f}\right) \\ \gamma_2 &= \left(0,\, \tfrac{2e-e\beta_2}{f}\right) \end{aligned} \tag{C.68}$$

APPENDIX D

Symmetry-Preserving Offsets

The following is a table of the symmetry-preserving offsets for each Bravais lattice expressed in terms of fractions of the lattice vectors.

Simple Cubic	$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$
Face Centered Cubic	$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$
Body Centered Cubic	None
Hexagonal	$\left(0, 0, \frac{1}{2}\right)$
Rhombohedral	$\left(0, 0, \frac{1}{2}\right)$
Simple Tetragonal	$\begin{pmatrix} 0, 0, \frac{1}{2} \\ \frac{1}{2}, \frac{1}{2}, 0 \\ \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \end{pmatrix}$
Body Centered Tetragonal	$\left(0, 0, \frac{1}{2}\right)$
Simple Orthorhombic	$\begin{pmatrix} 0, 0, \frac{1}{2} \\ 0, \frac{1}{2}, 0 \\ \frac{1}{2}, 0, 0 \\ 0, \frac{1}{2}, \frac{1}{2} \\ \frac{1}{2}, 0, \frac{1}{2} \\ \frac{1}{2}, \frac{1}{2}, 0 \\ \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \end{pmatrix}$
Base Centered Orthorhombic	$\begin{pmatrix} 0, 0, \frac{1}{2} \\ 0, \frac{1}{2}, 0 \\ 0, \frac{1}{2}, \frac{1}{2} \end{pmatrix}$
Face Centered Orthorhombic	$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$
Body Centered Orthorhombic	$\begin{pmatrix} 0, 0, \frac{1}{2} \\ 0, \frac{1}{2}, 0 \\ \frac{1}{2}, 0, 0 \end{pmatrix}$

Simple Monoclinic	$\begin{pmatrix} 0,\ 0,\ \frac{1}{2} \\ 0,\ \frac{1}{2},\ 0 \\ \frac{1}{2},\ 0,\ 0 \\ 0,\ \frac{1}{2},\ \frac{1}{2}, \\ \frac{1}{2}\ 0,\ \frac{1}{2}, \\ \frac{1}{2},\ \frac{1}{2},\ 0 \\ \frac{1}{2},\ \frac{1}{2},\ \frac{1}{2} \end{pmatrix}$
Base Centered Monoclinic	$\begin{pmatrix} 0,\ 0,\ \frac{1}{2} \\ 0,\ \frac{1}{2},\ 0 \\ -\frac{1}{4},\ \frac{1}{2},\ 0 \\ -\frac{1}{4},\ \frac{1}{4},\ \frac{1}{2} \\ \frac{1}{4},\ \frac{1}{4},\ 0 \\ \frac{1}{4},\ \frac{1}{4},\ \frac{1}{2} \\ 0,\ \frac{1}{2},\ \frac{1}{2} \end{pmatrix}$
Triclinic	None

APPENDIX E

Paper Copyright Licences

The following are licences or permissions to redistribute the publications inluded in this thesis. They appear in the following order:

1. Numerical Algorithm for Pòlya Enumeration Theorem
2. Generating derivative superstructures for systems with high configurational freedom
3. Efficiency of Generalized Regular **k**-point grids

Copyright Clearance Center

Confirmation Number: 11788824
Order Date: 02/05/2019

Customer Information

Customer: Wiley Morgan
Account Number: 3001395849
Organization: Wiley Morgan
Email: wiley.s.morgan@gmail.com
Phone: +1 (385) 325-0073
Payment Method: Invoice

This is not an invoice

Order Details

ACM journal of experimental algorithmics

Billing Status:
N/A

Order detail ID:	71797756
ISSN:	1084-6654
Publication Type:	e-Journal
Volume:	
Issue:	
Start page:	
Publisher:	Association for computing Machinery, Inc.
Author/Editor:	Association for Computing Machinery

Permission Status: ✓ **Granted**

Permission type: Republish or display content
Type of use: Republish in a thesis/dissertation
Order License Id: 4522730757010

Requestor type	Academic institution
Format	Print, Electronic
Portion	chapter/article
The requesting person/organization	Wiley S Morgan
Title or numeric reference of the portion(s)	The entire article.
Title of the article or chapter the portion is from	Numerical Algorithm for Pólya Enumeration Theorem
Editor of portion(s)	N/A
Author of portion(s)	Forcade, Rodney W. ; et al
Volume of serial or monograph	21
Issue, if republishing an article from a serial	1
Page range of portion	
Publication date of portion	Aug 17, 2016
Rights for	Main product
Duration of use	Life of current and all future editions
Creation of copies for the disabled	no
With minor editing privileges	no
For distribution to	Worldwide
In the following language(s)	Original language of publication

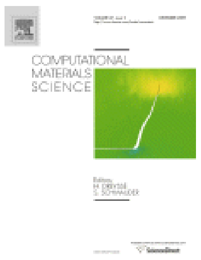
With incidental promotional use	no
Lifetime unit quantity of new product	Up to 499
Title	Using Symmetry to Accelerate Materials Discovery
Institution name	Brigham Young University
Expected presentation date	Feb 2019

Note: This item was invoiced separately through our **RightsLink service.** More info **$ 0.00**

Total order items: 1 **Order Total: $0.00**

Copyright Clearance Center RightsLink®

Home | Account Info | Help



Title: Generating derivative superstructures for systems with high configurational freedom

Author: Wiley S. Morgan,Gus L.W. Hart,Rodney W. Forcade

Publication: Computational Materials Science

Publisher: Elsevier

Date: August 2017

© 2017 Elsevier B.V. All rights reserved.

Logged in as:
Wiley Morgan
Account #:
3001395849

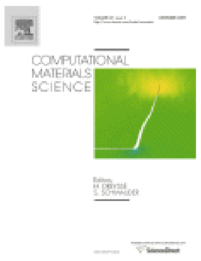
LOGOUT

Please note that, as the author of this Elsevier article, you retain the right to include it in a thesis or dissertation, provided it is not published commercially. Permission is not required, but please ensure that you reference the journal as the original source. For more information on this and on your other retained rights, please visit: https://www.elsevier.com/about/our-business/policies/copyright#Author-rights

BACK | CLOSE WINDOW

Copyright Clearance Center RightsLink®

Home | Account Info | Help



Title:	Efficiency of Generalized Regular k -point grids
Author:	Wiley S. Morgan,Jeremy J. Jorgensen,Bret C. Hess,Gus L.W. Hart
Publication:	Computational Materials Science
Publisher:	Elsevier
Date:	October 2018

Logged in as:
Wiley Morgan
Account #:
3001395849
LOGOUT

Please note that, as the author of this Elsevier article, you retain the right to include it in a thesis or dissertation, provided it is not published commercially. Permission is not required, but please ensure that you reference the journal as the original source. For more information on this and on your other retained rights, please visit: https://www.elsevier.com/about/our-business/policies/copyright#Author-rights

BACK | CLOSE WINDOW

Bibliography

[1] G. L. W. Hart and R. W. Forcade, “Algorithm for generating derivative structures,” Phys. Rev. B **77,** 224115 (2008).

[2] J. A. C. Weideman, “Numerical integration of periodic functions: A few examples,” The American mathematical monthly **109,** 21–36 (2002).

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

[4] P. Hohenberg and W. Kohn, “Inhomogeneous Electron Gas,” Phys. Rev. **136,** B864–B871 (1964).

[5] W. Kohn and L. J. Sham, “Self-Consistent Equations Including Exchange and Correlation Effects,” Phys. Rev. **140,** A1133–A1138 (1965).

[6] S. Curtarolo *et al.*, “AFLOW: an automatic framework for high-throughput materials discovery,” Comput. Mat. Sci. **58,** 218–226 (2012).

[7] J. E. Saal, S. Kirklin, M. Aykol, B. Meredig, and C. Wolverton, “Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD),” JOM **65,** 1501–1509 (2013).

[8] A. Jain *et al.*, “Commentary: The Materials Project: A materials genome approach to accelerating materials innovation,” APL Mat. **1,** 011002 (2013).

[9] S. L. Digabel, C. Tribes, and C. Audet, “NOMAD user guide,” Technical Report No. G-2009-37, Les cahiers du GERAD, Quebec, Canada (2009) .

[10] D. D. Landis, J. S. Hummelshøj, S. Nestorov, J. Greeley, M. Dułak, T. Bligaard, J. K. Nørskov, and K. W. Jacobsen, “The computational materials repository,” Comput. Sci. Eng. **14,** 51–57 (2012).

[11] J. Hachmann, R. Olivares-Amaya, S. Atahan-Evrenk, C. Amador-Bedolla, R. S. Sánchez-Carrera, A. Gold-Parker, L. Vogt, A. M. Brockway, and A. Aspuru-Guzik, “The Harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid,” J. Phys. Chem. Lett. **2,** 2241–2251 (2011).

[12] J. S. Hummelshøj, F. Abild-Pedersen, F. Studt, T. Bligaard, and J. K. Nørskov, “CatApp: a web application for surface chemistry and heterogeneous catalysis,” Angewandte Chemie **124,** 278–280 (2012).

[13] M. De Jong, W. Chen, H. Geerlings, M. Asta, and K. A. Persson, “A database to enable discovery and design of piezoelectric materials,” Sci. Data 2 (2015).

[14] M. De Jong *et al.*, “Charting the complete elastic properties of inorganic crystalline compounds,” Sci. Data **2,** 150009 (2015).

[15] L. Cheng, R. S. Assary, X. Qu, A. Jain, S. P. Ong, N. N. Rajput, K. Persson, and L. A. Curtiss, “Accelerating electrolyte discovery for energy storage with high-throughput screening,” J. Phys. Chem. Lett. **6,** 283–291 (2015).

[16] R. Gómez-Bombarelli *et al.*, “Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach,” Nat. Mater. **15,** 1120–1127 (2016).

[17] E. M. Chan, “Combinatorial approaches for developing upconverting nanomaterials: high-throughput screening, modeling, and applications,” Chem. Soc. Rev. **44,** 1653–1679 (2015).

[18] T. Tada, S. Takemoto, S. Matsuishi, and H. Hosono, “High-throughput ab initio screening for two-dimensional electride materials,” Inorg. Chem. **53,** 10347–10358 (2014).

[19] G. Pilania, C. Wang, X. Jiang, S. Rajasekaran, and R. Ramprasad, “Accelerating materials property predictions using machine learning,” Sci. Rep. 3 (2013).

[20] J. Yan, P. Gorai, B. Ortiz, S. Miller, S. A. Barnett, T. Mason, V. Stevanović, and E. S. Toberer, “Material descriptors for predicting thermoelectric performance,” Energy Environ. Sci. **8,** 983–994 (2015).

[21] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, “Quantum chemistry structures and properties of 134 kilo molecules,” Sci. Data **1,** 140022 (2014).

[22] J. Hachmann *et al.*, “Lead candidates for high-performance organic photovoltaics from high-throughput quantum chemistry–the Harvard Clean Energy Project,” Energy Environ. Sci. **7,** 698–704 (2014).

[23] L.-C. Lin *et al.*, "In silico screening of carbon-capture materials," Nat. Mater. **11,** 633–641 (2012).

[24] R. Armiento, B. Kozinsky, G. Hautier, M. Fornari, and G. Ceder, "High-throughput screening of perovskite alloys for piezoelectric performance and thermodynamic stability," Phys. Rev. B **89,** 134103 (2014).

[25] O. Senkov, J. Miller, D. Miracle, and C. Woodward, "Accelerated exploration of multi-principal element alloys with solid solution phases," Nat. Commun. 6 (2015).

[26] J. Greeley, T. F. Jaramillo, J. Bonde, I. Chorkendorff, and J. K. Nørskov, "Computational high-throughput screening of electrocatalytic materials for hydrogen evolution," Nat. Mater. **5,** 909 (2006).

[27] R. Gautier, X. Zhang, L. Hu, L. Yu, Y. Lin, T. O. Sunde, D. Chon, K. R. Poeppelmeier, and A. Zunger, "Prediction and accelerated laboratory discovery of previously unknown 18-electron ABX compounds," Nat. Chem. **7,** 308–316 (2015).

[28] A. O. Oliynyk and A. Mar, "Discovery of Intermetallic Compounds from Traditional to Machine-Learning Approaches," Accounts of chemical research (2017).

[29] H. Chen, G. Hautier, A. Jain, C. Moore, B. Kang, R. Doe, L. Wu, Y. Zhu, Y. Tang, and G. Ceder, "Carbonophosphates: a new family of cathode materials for Li-ion batteries identified computationally," Chem. Mat. **24,** 2009–2016 (2012).

[30] G. Hautier, A. Jain, S. P. Ong, B. Kang, C. Moore, R. Doe, and G. Ceder, "Phosphates as lithium-ion battery cathodes: an evaluation based on high-throughput ab initio calculations," Chem. Mater. **23,** 3495–3508 (2011).

[31] C. Jähne, C. Neef, C. Koo, H.-P. Meyer, and R. Klingeler, "A new LiCoPO 4 polymorph via low temperature synthesis," J. Mater. Chem. A **1,** 2856–2862 (2013).

[32] T. Moot, O. Isayev, R. W. Call, S. M. McCullough, M. Zemaitis, R. Lopez, J. F. Cahoon, and A. Tropsha, "Material informatics driven design and experimental validation of lead titanate as an aqueous solar photocathode," Materials Discovery **6,** 9–16 (2016).

[33] U. Aydemir *et al.*, "YCuTe 2: a member of a new class of thermoelectric materials with CuTe 4-based layered structure," J. Mat. Chem. A **4,** 2461–2472 (2016).

[34] H. Zhu *et al.*, "Computational and experimental investigation of TmAgTe 2 and XYZ 2 compounds, a new group of thermoelectric materials identified by first-principles high-throughput screening," J. Mat. Chem. C **3,** 10554–10565 (2015).

[35] W. Chen *et al.*, "Understanding thermoelectric properties from high-throughput calculations: trends, insights, and comparisons with experiment," J. Mat. Chem. C **4,** 4414–4426 (2016).

[36] G. Ceder, Y.-M. Chiang, D. Sadoway, M. Aydinol, Y.-I. Jang, and B. Huang, "Identification of cathode materials for lithium batteries guided by first-principles calculations," Nature **392,** 694–696 (1998).

[37] F. Yan, X. Zhang, G. Y. Yonggang, L. Yu, A. Nagaraja, T. O. Mason, and A. Zunger, "Design and discovery of a novel half-Heusler transparent hole conductor made of all-metallic heavy elements," Nat. Commun. 6 (2015).

[38] D. Bende, F. R. Wagner, O. Sichevych, and Y. Grin, "Chemical Bonding Analysis as a Guide for the Preparation of New Compounds: The Case of VIrGe and HfPtGe," Angewandte Chemie **129,** 1333–1338 (2017).

[39] A. Mannodi-Kanakkithodi, A. Chandrasekaran, C. Kim, T. D. Huan, G. Pilania, V. Botu, and R. Ramprasad, "Scoping the polymer genome: A roadmap for rational polymer dielectrics design and beyond," Mater. Today (2017).

[40] S. Sanvito, C. Oses, J. Xue, A. Tiwari, M. Zic, T. Archer, P. Tozman, M. Venkatesan, M. Coey, and S. Curtarolo, "Accelerated discovery of new magnets in the Heusler alloy family," Sci. Adv. **3,** e1602241 (2017).

[41] H. Yaghoobnejad Asl and A. Choudhury, "Combined theoretical and experimental approach to the discovery of electrochemically active mixed polyanionic phosphatonitrates, AFePO4NO3 (A= NH4/Li, K)," Chem. Mater. **28,** 5029–5036 (2016).

[42] G. Hautier, A. Miglio, G. Ceder, G.-M. Rignanese, and X. Gonze, "Identification and design principles of low hole effective mass p-type transparent conducting oxides," Nat. Commun. **4,** 2292 (2013).

[43] A. Bhatia *et al.*, "High-mobility bismuth-based transparent p-type oxide from high-throughput material screening," Chem. Mater. **28,** 30–34 (2015).

[44] G. H. Johannesson, T. Bligaard, A. V. Ruban, H. L. Skriver, K. W. Jacobsen, and J. K. Nørskov, “Combined electronic structure and evolutionary search approach to materials design,” Phys. Rev. Lett. **88,** 255506 (2002).

[45] D. P. Stucke and V. H. Crespi, “Predictions of new crystalline states for assemblies of nanoparticles: perovskite analogues and 3-D arrays of self-assembled nanowires,” Nano Lett. **3,** 1183–1186 (2003).

[46] S. Curtarolo, D. Morgan, and G. Ceder, “Accuracy of ab initio methods in predicting the crystal structures of metals: A review of 80 binary alloys,” Calphad **29,** 163–211 (2005).

[47] S. F. Matar, I. Baraille, and M. Subramanian, “First principles studies of SnTiO 3 perovskite as potential environmentally benign ferroelectric material,” Chem. Phys. **355,** 43–49 (2009).

[48] G. Ceder, G. Hautier, A. Jain, and S. P. Ong, “Recharging lithium battery research with first-principles methods,” MRS Bulletin **36,** 185–191 (2011).

[49] A. N. Sokolov *et al.*, “From computational discovery to experimental characterization of a high hole mobility organic crystal,” Nat. Commun. **2,** 437 (2011).

[50] Z. W. Ulissi *et al.*, “Machine-Learning Methods Enable Exhaustive Searches for Active Bimetallic Facets and Reveal Active Site Motifs for CO2 Reduction,” ACS Catal. **7,** 6600–6608 (2017).

[51] O. Levy, R. V. Chepulskii, G. L. Hart, and S. Curtarolo, “The new face of rhodium alloys: revealing ordered structures from first principles,” JACS **132,** 833–837 (2009).

[52] X. Ma, G. Hautier, A. Jain, R. Doe, and G. Ceder, “Improved capacity retention for LiVO2 by Cr substitution,” J. Electrochem. Soc. **160,** A279–A284 (2013).

[53] K. Yang, W. Setyawan, S. Wang, M. B. Nardelli, and S. Curtarolo, “A search model for topological insulators with high-throughput robustness descriptors,” Nat. Mater. **11,** 614–619 (2012).

[54] H. Chen, G. Hautier, and G. Ceder, “Synthesis, computed stability, and crystal structure of a new family of inorganic compounds: carbonophosphates,” JACS **134,** 19619–19627 (2012).

[55] S. Kirklin, B. Meredig, and C. Wolverton, “High-Throughput Computational Screening of New Li-Ion Battery Anode Materials,” Advanced Energy Materials **3,** 252–262 (2013).

[56] V. L. Deringer and G. Csányi, “Machine learning based interatomic potential for amorphous carbon,” Physical Review B **95,** 094203 (2017).

[57] D. Dragoni, T. D. Daff, G. Csányi, and N. Marzari, “Achieving DFT accuracy with a machine-learning interatomic potential: Thermomechanics and defects in bcc ferromagnetic iron,” Physical Review Materials **2,** 013808 (2018).

[58] A. Shapeev, “Accurate representation of formation energies of crystalline alloys with many components,” Computational Materials Science **139,** 26–30 (2017).

[59] J. W. Yeh, S. K. Chen, S. J. Lin, J. Y. Gan, T. S. Chin, T. T. Shun, C. H. Tsau, and S. Y. Chang, “Nanostructured High-Entropy Alloys with Multiple Principal Elements: Novel Alloy Design Concepts and Outcomes,” Advanced Engineering Materials **6,** 299–303 (2004).

[60] Y. Zhang, T. T. Zuo, Z. Tang, M. C. Gao, K. A. Dahmen, P. K. Liaw, and Z. P. Lu, “Microstructures and properties of high-entropy alloys,” Progress in Materials Science **61,** 1–93 (2014).

[61] M. C. Troparevsky, J. R. Morris, P. R. C. Kent, A. R. Lupini, and G. M. Stocks, “Criteria for Predicting the Formation of Single-Phase High-Entropy Alloys,” Physical Review X **5,** 011041 (2015).

[62] D. Miracle and O. Senkov, “A critical review of high entropy alloys and related concepts,” Acta Materialia **122,** 448 – 511 (2017).

[63] K. Lejaeghere, V. Van Speybroeck, G. Van Oost, and S. Cottenier, “Error estimates for solid-state density-functional theory predictions: an overview by means of the ground-state elemental crystals,” Critical Reviews in Solid State and Materials Sciences **39,** 1–24 (2014).

[64] M.-C. Kim, E. Sim, and K. Burke, “Understanding and reducing errors in density functional calculations,” Physical review letters **111,** 073003 (2013).

[65] P. D’Arco, S. Mustapha, M. Ferrabone, Y. Noël, M. De La Pierre, and R. Dovesi, “Symmetry and random sampling of symmetry independent configurations for the simulation of disordered solids,” J. Phys. Condens. Matt. **25,** 355401 (2013).

[66] A. Van De Walle and G. Ceder, “Automating first-principles phase diagram calculations,” J Phase Equilib. **23,** 348 (2002).

[67] A. Van De Walle, M. Asta, and G. Ceder, “The alloy theoretic automated toolkit: A user guide,” Calphad **26,** 539–553 (2002).

[68] N. A. Zarkevich, T. L. Tan, and D. D. Johnson, “First-principles prediction of phase-segregating alloy phase diagrams and a rapid design estimate of their transition temperatures,” Phys. Rev. B **75,** 104203 (2007).

[69] G. L. Hart, L. J. Nelson, and R. W. Forcade, “Generating derivative structures at a fixed concentration,” Comput. Mater Sci. **59,** 101 – 107 (2012).

[70] G. L. W. Hart and R. W. Forcade, “Generating derivative structures from multilattices: Algorithm and application to hcp alloys,” Phys. Rev. B **80,** 014120 (2009).

[71] M. J. Buerger, “Derivative Crystal Structures,” J. Chem. Phys. **15,** 1 (1947).

[72] R. Grau-Crespo and S. Hamad, “The symmetry-adapted configurational ensemble approach to the computer simulation of site-disordered solids,” In *MOL2NET, International Conference on Multidisciplinary Sciences*, p. c002 (MDPI, Basel, Switzerland, 2015).

[73] J. Moreno and J. M. Soler, “Optimal meshes for integrals in real- and reciprocal-space unit cells,” Phys. Rev. B **45,** 13891–13898 (1992).

[74] P. Wisesa, K. A. McGill, and T. Mueller, “Efficient generation of generalized Monkhorst-Pack grids through the use of informatics,” Phys. Rev. B **93,** 155109 (2016).

[75] G. Kresse and J. Hafner, “Ab initio molecular dynamics for liquid metals,” Phys. Rev. B **47,** 558 (1993).

[76] P. Q. Nguyen and D. Stehlé, “Low-Dimensional Lattice Basis Reduction Revisited,” In *Algorithmic Number Theory*, pp. 338–357 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2004).

[77] I. Křivý and B. Gruber, “A Unified Algorithm for Determining the Reduced (Niggli) Cell,” Acta Crystallogr. A 32 (1976).

[78] A. Santoro and A. D. Mighell, “Determination of reduced cells,” Acta Crystallogr. A **26,** 124–127 .

[79] R. W. Grosse-Kunstleve, N. K. Sauter, and P. D. Adams, “Numerically stable algorithms for the computation of reduced unit cells,” Acta Crystallogr. A **60,** 1–6 (2004).

[80] edited by Theo Hahn, *International tables for crystallography. Volume A, Space-group symmetry* (Fifth, revised edition. Dordrecht ; London : Published for the International Union of Crystallography by Kluwer Academic Publishers, 2002., 2002).

[81] G. L. W. Hart, J. Jorgensen, W. M. Morgan, and R. W. Forcade, “A robust algorithm for **k**-point grid generation and symmetry reduction,” .

[82] G. Kresse and J. Furthmüller, “Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set,” Comput. Mat. Sci. **6,** 15–50 (1996).

[83] G. Kresse and J. Hafner, “Ab initio molecular-dynamics simulation of the liquid-metal–amorphous-semiconductor transition in germanium,” Phys. Rev. B **49,** 14251 (1994).

[84] G. Kresse and J. Furthmüller, “Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set,” Phys. Rev. B **54,** 11169 (1996).

[85] P. E. Blöchl, “Projector augmented-wave method,” Phys. Rev. B **50,** 17953 (1994).

[86] G. Kresse and D. Joubert, “From ultrasoft pseudopotentials to the projector augmented-wave method,” Phys. Rev. B **59,** 1758 (1999).