A Fast, Parallel Algorithm for Distant-dependent Calculation of Crystal Properties

Matthew Stein
Southern Methodist University
3215 Daniel Ave.
Dallas, Texas, 75205, USA
(Dated: September 6, 2017)

A fast, parallel algorithm for distant-dependent calculation and simulation of crystal properties is presented along with speedup results and methods of application. An illustrative example is used to compute the Lennard-Jones lattice constants up to 32 significant figures for $4 \leq p \leq 30$ in the simple cubic, face-centered cubic, body-centered cubic, hexagonal-close-pack, and diamond lattices. In most cases, the known precision of these constants is more than doubled, and in some cases, corrected from previously published figures. The tools and strategies to make this computation possible are detailed along with application to other potentials, including those that model defects.

Keywords: Parallel Algorithms, Crystal Potential Energy, Lattice Constants, Computational Approaches, Classical Potentials

I. INTRODUCTION

Calculations of crystal potentials or force interactions, whether through molecular dynamics or classical potentials, will rely on functions of distances between many atoms. In either case, computational complexity and time will limit the precision with which values are calculated. Even in the case of classical potentials, which are less computationally intense, crystal simulations and calculations are usually limited to the millions of atoms, with determined values often having fewer significant figures than a single-precision float.

Classical potential fitting has also become more complex in attempts to adapt a single model to a greater number of situations. The Lennard-Jones potential is simple and widely used for its computational speed, but much more accurate models exist. The Buckingham potential expanded on the Lennard-Jones potential, replacing the Pauli repulsive term with an exponential function but at computational cost. The Stillinger-Weber potential (hereafter SW potential) was proposed as a further improvement, now taking into account not just the distance between atoms but also the angles of their bonds in a new 3-body term.

Improvements on the classical potentials have thus progressed for decades, with attempts to find a potential model that works not only with perfects crystals, but those with point defects, plane defects, and more. A fitted formula in one situation (temperature, lattice, atomic composition) often does not suitably agree with experimental values from another. As such, the potentials grow ever more complex, and determining parameters comes at greater cost, but the objective of a transferable model remains a priority.

Rather than limiting calculations to a small number of atoms (and thus limited precision), or expanding compute time (which schedules and resources may not permit), a faster optimized algorithm could be used to achieve better and/or less costly results. Additionally, potentials with arbitrary cut-off values (often used to shorten compute time) can be relaxed for better fitting of other parameters and more realistic simulation. An adaptive algorithm would also ideally be suited for studies of non-ideal lattices with defects, vacancies or other imperfections.

The inclusion of contributions from further atoms or those with defect locations should also come with questions about the precision of the calculation. For example, a single interstitial sufficiently far away from a reference atom may not affect the total potential energy, but a plane defect at the same distance may have significant contributions when all atoms across the plane are considered. It may be useful to use very high-precision variables in computation, further advancing the need for a faster algorithm.

II. COMPUTATIONAL APPROACH

Potential and force calculations in a crystal depend on distances between pairs of atoms. Any summation over lattice points will first require the calculation of the distance between these atoms $r_{ij}$, and then apply some function $f(r_{ij})$ to that distance. The return value is included in the total sum. The algorithms presented here can be used for any such distance-dependent function.

For illustrative purposes, the Lennard-Jones potential will be used as an example of the computational power of this new algorithm. Further extensions and adaptations of the same algorithm to other functions and potentials are discussed in Section III.

The author would like to note there are many common techniques to optimize algorithms, especially nested loops, such as avoiding the repetitive calculation of the same value. Likewise there are algorithms to avoid round-off error such as the Kahan summation algorithm. These common tools are omitted from the algorithms presented here to more clearly show the logic structure, and to more clearly demonstrate what new methods are
The Lennard-Jones potential is a simple but widely-used potential energy formula. The total potential energy of a crystal with \( N \) atoms is described by the sum of Equation (1) between all pairs of atoms. The constant parameters \( \sigma \) and \( \epsilon \) are determined from experimental measurements, and \( d_j \) is the distance from a fixed reference atom to any other atom \( j \) as a multiple of the nearest-neighbor distance.

\[
U_{\text{tot}} = 2N\epsilon \left[ \sum_{j=1}^{\infty} \left( \frac{\sigma}{d_j} \right)^{12} - 2 \sum_{j=1}^{\infty} \left( \frac{\sigma}{d_j} \right)^{6} \right] \tag{1}
\]

To simplify calculations, it is useful to separate the \( d_j \) terms and examine them independently:

\[
L_p = \sum_{j=1}^{\infty} \left( \frac{1}{d_j} \right)^p \tag{2}
\]

It is seen that Equation (1) can be determined by first calculating these lattice constants \( L_p \) for \( p = 6 \) and \( p = 12 \). The \( p = 6 \) term represents the attractive van der Waals force, whereas the Pauli exclusion principle is responsible for the repulsive \( p = 12 \) term. The choice of \( p = 12 \) is not fully motivated from first principles, so it is useful to compute a range of \( p \) values. For \( p < 4 \), the series does not converge, and for \( p > 30 \), the series is seen to converge to the coordination number of the lattice. While any real value of \( p \) could be computed, this example uses integer values for comparison to other published results which also examine integer values of \( p \).

To achieve a useful value of the lattice constants \( L_p \) in Equation (2), the series need only converge to the precision required. The double-precision float has \( \sim 15 \) decimal digits, and is now a very fast variable to use with most modern compilers. Results have been published for the simple cubic (SC), face-centered cubic (FCC), body-centered cubic (BCC), hexagonal-close-pack (HCP) lattices with up to 15 decimal digits, but not every term published has actually converged to the precision given, especially for \( p < 12 \). The diamond (DIA) lattice has been published up to 9 decimal digits, roughly the precision of a 32-bit single-precision float. To fully demonstrate the power of the algorithms in this work, the Portable, Extensible Toolkit for Scientific Computation (PETS) was used to implement 128-bit floats to push the precision to 32 decimal digits.

### Algorithm 1 Brute Force Method

\[
L_p = 0
\]

for \( X \leftarrow -(D/2) \) to \((D/2)\) do

for \( Y \leftarrow -(D/2) \) to \((D/2)\) do

for \( Z \leftarrow -(D/2) \) to \((D/2)\) do

if \( X = 0 \) and \( Y = 0 \) and \( Z = 0 \) then

\[
L_p += \frac{1}{(X^2 + Y^2 + Z^2)^{p/2}}
\]

else

Next

end if

end for

end for

end for

return \( L_p \)

The if-statement is present to avoid the \( \frac{1}{0} \) term (at the origin) which would otherwise set \( L_p \) equal to infinity or \( NaN \). At this point, knowing that there will be \( (D+1)^3 \) if-statements checked in every run of Algorithm it is worth finding how many terms will be necessary for this sum to converge.

### C. The Convergent Series

Depending on implementation of 128-bit floats, these variables yield \( \sim 32 \) decimal digits for each term. Finding where Equation (2) converges then requires additional terms to be equal to or less than \( 10^{-33} \) (in arbitrary units). Finding the coordinates of where Equation (2) with respect to \( Y \) and \( Z \) for \( p = 6 \) and \( X = R \). That result is multiplied by 6 for symmetry.
While an exact result requires the actual summation in Equation (2), this result is useful for determining how many terms are required for convergence to a particular precision.

\[
\sum_{\text{face} \in R} p, R \propto \frac{1}{R^{(p-2)}}
\]

\[
\sum_{\text{face} \in R} = 6 \times \frac{2 + 15\sqrt{2}\text{ArcCot}\sqrt{2}}{12R^4} \approx \frac{7.52815}{R^4}
\]

FIG. 1. Average value of terms added to \( L_6 \) from the face at some fixed \( R \) versus the distance and number of terms added to \( L_6 \) at that face.

The convergence of Equation (2) is much faster for higher values of \( p \) (Figure 2) but presents a significant computational challenge for low \( p \). Converging to any desired precision at low \( p \) will then require finding fast algorithms that will capitalize on efficiency, parallelism, and any inherent symmetries in the crystal lattice.

D. Finding Speedup

1. Avoiding Unnecessary Operations

In the simple case of Algorithm 1, the \((D + 1)^3\) if statements can be avoided by structuring the program to calculate different regions of the same cube, none of which contain the \((0,0,0)\) position (Figure 3). There are now six regions to consider: two rectangular parallelepipeds, two planes, and two lines. The loops for these regions are executed in serial (Algorithm 2).

2. Parallelization

Since each individual \( L_{p_i} \) value is independent of every other \( L_{p_j} \), Algorithm 2 is an excellent candidate for parallelization via MPI. The parallelization of these nested for-loops, however, requires the following careful prescription such that each thread does approximately the same amount of work, and the entire \( 3 \)-dimensional grid of lattice points is covered. For \( \text{NumProcs} \) threads, one cannot simply set thread number \( \text{MyID} \) to cover
a range of \((D/\text{NumProcs})\) in \((X/Y/Z)\) as can be trivially done in the case of a 1-dimensional array. Instead, the original cube from Algorithm 1 is broken down into \(\text{NumProcs}\) inter-penetrating cubes with a different basis. This allows every thread to compute \(L_p\), the results of which can be combined at the end of the algorithm. The integer basis of each new lattice is computed as follows:

\[
\text{Basis} = \text{Floor}(\sqrt[3]{\text{NumProcs}}) \tag{4}
\]

The initial \((X,Y,Z)\) position of each thread is:

\[
\begin{align*}
X_i &= (MyID\%\text{Basis}) + (D/2) \\
Y_i &= \left(\text{Floor}\left[\frac{MyID}{\text{Basis}}\right]\%\text{Basis}\right) + (D/2) \tag{5} \\
Z_i &= \left(\text{Floor}\left[\frac{MyID}{\text{Basis}^2}\right]\%\text{Basis}\right) + (D/2)
\end{align*}
\]

Fortunately, only one thread (hereafter the origin thread) will pass through the \((0,0,0)\) position. All other threads can execute a fast triple-nested \texttt{for}-loop (Algorithm 3 Figure 4), and the origin thread will execute a slightly modified version of Algorithm 2. The origin thread is identified as:

\[
k = (D/2)\%\text{Basis}
\]

\[
\text{OriginThreadNum} = k * \text{Basis}^2 + k * \text{Basis} + k
\]
optional, erasing the need for programming more flexible methods.

3. Exploiting Symmetry

In the case of the SC lattice, the calculation of \( L_p \) can be shortened by considering that the cube is made of eight identical, smaller pieces corresponding to each octant. Therefore a speedup of almost eightfold can be found by calculating only one of these octants and multiplying the end result. However, the algorithmic range of each octant is not as obvious as it seems. There are unit cells along the planes between octants whose atoms need to have their contributions handled carefully as some of the atoms sit astride different octants (Figure 5), and likewise for cells along the axes. For unit cells immediately adjacent other octants, consider these as being in separate volumes called the axis or face volumes (for cells touching the axes or faces between octants, respectively) as in Figure 6. The remaining cells are considered to be in one of eight cubic volumes spanning the rest of each octant. Therefore, in the entire lattice, there are eight cubic volumes, 12 face volumes, and six half-axis volumes. In the case of the SC lattice, one need only calculate the sum of a single cubic volume \( L_{cube} \), a single face \( L_{face} \), and a single half-axis \( L_{axis} \) to determine \( L_p \) (Equation 6).

Algorithm 4 Symmatrized

\[
\begin{aligned}
L_{cube} &= L_{face} = L_{axis} = 0 \\
&\text{for } X \leftarrow (D/2) \text{ to } 1 \text{ do} \\
&\quad \text{for } Y \leftarrow (D/2) \text{ to } 1 \text{ do} \\
&\quad\quad \text{for } Z \leftarrow (D/2) \text{ to } 1 \text{ do} \\
&\quad\quad\quad L_{cube} += \frac{1}{(X^2 + Y^2 + Z^2)^{p/2}} \\
&\quad\quad\quad L_{face} += \frac{1}{(X^2 + Y^2)^{p/2}} \\
&\quad\quad\quad L_{axis} += \frac{1}{X^p} \\
\text{return } [L_{cube} + (1.5 \times L_{face}) + (0.75 \times L_{axis})] \times 8
\end{aligned}
\]

The values 1.5 and 0.75 arise in Equation 6 from the fact that there are 12 faces and six half-axes that should contribute equally to each of the eight octants, so \( \frac{12}{8} = 1.5 \) for the faces, and \( \frac{6}{8} = 0.75 \) for the axes. The new serial algorithm, which automatically avoids the origin, can be written compactly as in Algorithm 4.

\[
L_p = [L_{cube} + (1.5 \times L_{face}) + (0.75 \times L_{axis})] \times 8 \quad (6)
\]

To parallelize this, the basis is calculated as before, but the same basis does not hold for calculation of the face or axis (2- and 1-dimensional arrays, respectively). The calculation of those bases is shown in Algorithm 5.

When looping through each volume (cube, face, or axis as shown in Algorithm 4), the values of \((X_i, Y_i, Z_i)\) must be calculated relative to the appropriate basis for that volume (Equation 4 with \( Basis \), \( Basis_f \) or \( Basis_a \), as needed). Note that the step sizes must be negative since the initial positions are set at points away from the origin. The new algorithm is computed by all threads as there is no need to find an origin thread.
are cells where an atom sits astride the plane separating the octants. These atoms should only be counted once. However, there are other basis atoms in the volume of the unit cells immediate next to these planes (or around the axes) which need to be effectively counted once, but since the number of faces and axes that are shared is different from the number of octants, they need to be counted with special weights.

4. Extending the Exploitation of Symmetry: BCC and FCC

For the BCC and FCC lattices, the same exploitation of octants can be used, but with special handling: The lattices must be thought of as an SC lattice with two and four basis atoms, respectively. The for–loop variables now indicate the coordinates of the new conventional unit cells, instead of just the atoms. The nearest-neighbor distance must be normalized properly to this new conventional unit cell (Table 1), and the first triple-nested for–loop in Algorithm 7 can be computed similarly to the SC case, with the additional basis atom(s) added at each unit cell location. However, the face- and axis-cells are handled uniquely.

The multiple counts of basis atoms in Algorithm 7 are due to the way they are “shared” between the octants of the broken up cube. Along the face or axis, there

| Algorithm 5 Calculation of Basis |
|---------------------------------|
| $T = \text{NumProcs}$          |
| while $(\sqrt{T} - \text{Floor}(\sqrt{T})) > 0$ do |
| $T = T - 1$                    |
| $\text{Basis}_{f} = \text{Floor}(\sqrt{T})$          |
| $\text{Basis}_{a} = \text{NumProcs} - T$ |

| Algorithm 6 Parallelized & Symmetrized |
|----------------------------------------|
| $L_{\text{cube}} = L_{\text{face}} = L_{\text{axis}} = 0$ |
| // Cube Volume |
| ... calculate $X_i, Y_i, Z_i$ relative to Basis ... |
| for $X = X_i$ to $X > 0$ in steps of $-\text{Basis}_{f}$ do |
| for $Y = Y_i$ to $Y > 0$ in steps of $-\text{Basis}_{f}$ do |
| for $Z = Z_i$ to $Z > 0$ in steps of $-\text{Basis}_{f}$ do |
| $L_{\text{cube}} += \frac{1}{(X^2 + Y^2 + Z^2)^2}$ |
| // Face & Axis |
| if $\text{Basis}_{a} > 0$ then |
| if $\text{MyID} < T$ then |
| ... calculate $X_i, Y_i$ relative to Basis$_{f}$ ... |
| for $X = X_i$ to $X > 0$ in steps of $-\text{Basis}_{f}$ do |
| for $Y = Y_i$ to $Y > 0$ in steps of $-\text{Basis}_{f}$ do |
| $L_{\text{face}} += \frac{1}{(X^2 + Y^2)^2}$ |
| ... set $X_i$ as ProcID + D ... |
| for $X = X_i$ to $X > 0$ in steps of $-\text{NumProcs}$ do |
| $L_{\text{axis}} += \frac{1}{X^2}$ |
| else |
| ... calculate $X_i, Y_i$ relative to Basis$_{a}$ ... |
| for $X = X_i$ to $X > 0$ in steps of $-\text{Basis}_{a}$ do |
| $L_{\text{axis}} += \frac{1}{X^2}$ |
| else |
| ... calculate $X_i, Y_i$ relative to Basis$_{a}$ ... |
| for $X = X_i$ to $X > 0$ in steps of $-\text{Basis}_{a}$ do |
| for $Y = Y_i$ to $Y > 0$ in steps of $-\text{Basis}_{a}$ do |
| $L_{\text{face}} += \frac{1}{(X^2 + Y^2)^2}$ |
| ... set $X_i$ as ProcID + D ... |
| for $X = X_i$ to $X > 0$ in steps of $-\text{NumProcs}$ do |
| $L_{\text{axis}} += \frac{1}{X^2}$ |
| ... MPI summation ... |
| return $[L_{\text{cube}} + (1.5 \times L_{\text{face}}) + (0.75 \times L_{\text{axis}})] \times 8$ |

| Algorithm 7 FCC - Symmetrized |
|--------------------------------|
| $L_{\text{cube}} = L_{\text{face}} = L_{\text{axis}} = 0$ |
| $n = 2.0$ // $n = \text{Normalization factor}$ |
| // Basis atom offsets |
| $b_{x} = 0.5; b_{y} = 0.5; b_{z} = 0.0$ |
| $b_{x} = 0.0; b_{y} = 0.5; b_{z} = 0.5$ |
| $b_{x} = 0.5; b_{y} = 0.0; b_{z} = 0.5$ |
| for $X \leftarrow (D/2)$ to 1 do |
| if $Y \leftarrow (D/2)$ to 1 do |
| // First basis atom |
| $R = (X^2 + Y^2 + Z^2) * n$ |
| $L_{\text{cube}} += \frac{1}{R^2}$ |
| // Second basis atom |
| $R = ((X + b_{x})^2 + (Y + b_{y})^2 + (Z + b_{z})^2) * n$ |
| $L_{\text{cube}} += \frac{1}{R^2}$ |
| ... similarly for the other basis atoms ... |
| $R = (X^2 + Y^2) * n$ |
| $L_{\text{face}} += \frac{1}{R^2}$ |
| ... then count basis atom 2 once ... |
| ... then count basis atoms 3 & 4 twice ... |
| $R = (X^2) * n$ |
| $L_{\text{axis}} += \frac{1}{R^2}$ |
| ... then count basis atom 2 twice ... |
| ... then count basis atom 3 four times ... |
| ... then count basis atom 4 once ... |
| return $[L_{\text{cube}} + (1.5 \times L_{\text{face}}) + (0.75 \times L_{\text{axis}})] \times 8$ |
FIG. 7. Conventional FCC unit cells along the X-Y plane (red). For face terms, the shared atoms (yellow spheres) are counted once in Algorithm 7 whereas the mirrored atoms (solid blue spheres) must be counted twice to make full use of symmetry. The faded blue spheres indicate which atoms are mirroring those indicated by the solid blue atoms. Gray spheres indicate other FCC atoms in other unit cells along the red plane.

In the BCC structure, the same mirroring principle applies: the second basis atom is counted once for cells in the 3-dimensional volume spanned, twice for cells along the face, and four times for cells along the axis.

| Lattice | Basis Atoms | Normalization Factor |
|---------|-------------|----------------------|
| SC      | 1           | 1                    |
| BCC     | 2           | 2/4                  |
| FCC     | 4           | 2                    |
| DIA     | 12          | 12                   |

TABLE I. Number of basis atoms and normalization factors in the conventional unit cells for each lattice.

5. Extending the Exploitation of Symmetry: DIA

For diamond, the conventional unit cell is essentially an FCC conventional cell with the addition of four more basis atoms within the volume of the cell at the tetrahedral positions. The algorithm requires further special handling due to the asymmetry of the tetrahedral positions across one axis. In the BCC and FCC cases, the symmetry between octants obeyed rotational symmetry in that rotating the view 90 degrees about any axis resulted in viewing the exact same configuration of atoms. However, the diamond lattice does not have this symmetry. When rotating 90 degrees, the tetrahedral atoms now appear at different distances (Figure 8).

FIG. 8. Two conventional cells of the DIA lattice are shown. Distances to the tetrahedral atoms are not the same when rotating 90 degrees about any point. The blue spheres are the FCC-like basis atoms, and the red spheres indicate the tetrahedral atoms. The sizes of the spheres are not indicative of the sizes of the atoms at these sites.

Rather than settling for only four-fold speedup in symmetrization, one can fashion a new conventional unit cell that, while physically unrealistic, presents the same mathematical results as a real DIA lattice for this calculation. The new conventional unit cell has 12 basis atoms where four are the usual FCC-like atoms, four are the original tetrahedral atoms, and an additional four atoms occupy the location of where the tetrahedral atoms would appear to be if the viewer rotates 90 degrees (Figure 9).

FIG. 9. The 12-basis-atom conventional unit cell for calculations involving the diamond lattice. The orange spheres indicate the additional tetrahedral atoms.

All of the tetrahedral atom contributions to $L_p$ now need to be counted for half of what they normally would
since there are now twice as many (Table II). Doing so yields an identical mathematical result from any other approach, but allows for full eight-fold speedup by only calculating one octant. The exact weights for each basis atom in each volume of the algorithm are described in Table II.

| Basis Atom (X,Y,Z) | Cube Multiple | Face Multiple | Axis Multiple |
|--------------------|---------------|---------------|---------------|
| 1 (0,0,0)          | 1             | 1             | 1             |
| 2 (1/2,1/2,0)      | 1             | 1             | 2             |
| 3 (0,1/2,1/2)      | 1             | 2             | 4             |
| 4 (1/4,0,1/4)      | 1             | 2             | 2             |
| 5 (1/4,1/4,1/4)    | 0.5           | 1             | 2             |
| 6 (3/4,1/4,1/4)    | 0.5           | 1             | 2             |
| 7 (3/4,0,1/4)      | 0.5           | 1             | 2             |
| 8 (1/4,3/4,1/4)    | 0.5           | 1             | 2             |
| 9 (3/4,3/4,1/4)    | 0.5           | 1             | 2             |
| 10 (3/4,1/4,3/4)   | 0.5           | 1             | 2             |
| 11 (1/4,3/4,3/4)   | 0.5           | 1             | 2             |
| 12 (1/4,1/4,3/4)   | 0.5           | 1             | 2             |

TABLE II. Conventional DIA lattice basis atoms (2-8) and additional tetrahedral basis atoms (9-12) for use in Algorithms 4 and 6. The Cube Multiple represents the numerator used for the \((L_{cube}/face/axis += )\) lines in pseudocode.

6. Extending the Exploitation of Symmetry: HCP

For HCP, due to the hexagonal nature of the lattice, a completely different approach is used. Using the fact that the HCP lattice has alternating layers (ABABAB) and those layers have alternating and repeating rows, the structure can be logically constructed as four inter-penetrating orthorhombic sub-lattices. One sub-lattice must be chosen to contain the \((0,0,0)\) position, whereas the others are identical in shape but offset from this first sub-lattice (Figure 10). An algorithm can be constructed to calculate these four sub-lattices separately, and each can be parallelized as before.

The symmetry of this lattice can be easily broken down into quartets, but using octants will present a similar challenge as the DIA lattice. One of the four sub-lattices will extend slightly beyond what would be one of the faces between octants (Figure 11), and distances to each atom from the origin are not the same across this axis. The other three sub-lattices have atoms that either lie exactly on the faces, or completely within an octant. The solution, similar to DIA, is to double the number of atoms in the only sub-lattice with unequally shared atoms. The positions of the extra atoms will be those that respect the rotational symmetry required for splitting the entire HCP lattice into equal octants. As with DIA, the algorithm halves the value added to \(L_p\) from each atom in this sub-lattice.

7. Onionization

Running large, parallelized jobs on a cluster is convenient for solving large problems such as the algorithms described above. However, software and hardware errors do occasionally occur which can result in many lost CPU hours. As such, it is wise to break one large computation into many small ones. The result is a series of jobs that stack like layers of a (cubic) onion that are gradually added to the problem set (Figure 12, Left). This has the added benefit of being able to reduce roundoff error for extremely small terms (i.e. those layers at greatest distance) if one performs the sum of each job’s return value from smallest to greatest.

In the symmatrized version of the program, the expedient use of this method would involve breaking the
onion layer down into six new volumes: three volumes that span the main cube volume from the inner layer to the new outer layer, two regions to cover the face, and one region to cover the axis (Figure 12, Right). This avoids having to check that the coordinates covered are outside of the previous layer, and the cost of entering and leaving the for-loops is negligible compared to the number of if-statements avoided. Fortunately, the calculation of bases for each volume are identical to the non-onionized version.

III. RESULTS & APPLICATIONS

A. Lennard-Jones Lattice Constants

Using the symmetrized and parallelized algorithms described above, the Lennard-Jones lattice constants \( L_p \) have been calculated in the SC, BCC, FCC, HCP, and DIA lattices (Table IV). Terms with \( p > 9 \) are computed to 32 decimal digits, convenient for quadruple precision calculations. Those with \( p \leq 9 \) are computed to lower precision due to computational limits (Figure 2). In addition to extending the precision of these constants, there are corrections to terms with \( p < 12 \) previously published \([10, 2]\). The total speedup achieved going from the brute force method to the symmetric, parallel program was \( \sim 29 \) fold (Table III).

To validate these results, a Mathematica program similar to Algorithm 2 was used for several terms with \( p \geq 12 \) using infinite precision in all five lattices. Higher order terms were chosen because of the faster convergence of higher \( p \) values, and the comparative slowness of Algorithm 2 with the use of infinite precision. The results all agreed to the given precision in Table IV.

Moreover, a new method of computing any value that depends on distance between atoms in a crystal lattice has been created and optimized. This same algorithm can be tailored to look at other crystal energy functions, such as the Buckingham potential, SW potential, and others.

| Algorithm | Fraction of \( D^3 \) Effective Speedup terms in \( L_p \) per CPU core |
|-----------|---------------------------------------------------------------|
| Simple 1  | 1.00                                                          |
| Broken Down 2 | 2.671                                                        |
| Parallel 3 | 3.476                                                        |
| Symmetric Parallel 4 | 0.1249, 28.99                                                  |

B. Other Classical Potentials

As the results in Table III are normalized to 1, similar speed-up values should be attainable for applications of this approach to other crystal calculations. For example, the SW potential is fit with as many as nine parameters:

\[
U_{tot} = \epsilon \sum_{i<j} f_2(r_{ij}/\sigma) + \sum_{i<j<k} f_3(r_{ij}/\sigma, r_{ik}/\sigma, \theta_{ijk})
\]

\[
f_2(r) = \begin{cases} A(Br^{-p} - r^{-q}) \exp[(r - a)^{1}], & r < a \\ 0, & r \geq a \end{cases}
\]

\[
f_3(r_{ij}, r_{ik}, \theta_{ijk}) = \lambda \left( \cos \theta_{ijk} + \frac{1}{3} \right)^2 \exp \left( \frac{\gamma}{r_{ij} - a} \right) \times \exp \left( \frac{\gamma}{r_{ik} - a} \right)
\]

Fitting these parameters over many lattice sites and simulation requires lengthy computation, but it is also the case that the range of the potential is cut off at some arbitrary value (in this case, \( a \)). Indeed the cutoff is typically so short that only nearest- or next-to-nearest neighbors contribute to the total energy. Relaxing this parameter would allow simulation of more effects from vacancy or interstitial events. The algorithms described above can be used to compensate for the additional calculations, resulting in a potentially more transferable fit.

C. Applications to Crystal Defects

To simulate defects, one cannot use an algorithm for calculating over lattice sites in a perfect crystal. For example, to test or fit parameters for the creation energy...
of a Frenkel pair \cite{14}, the atom at the origin in the algorithms in this paper can be instead walked along (or integrated over) the path of defect creation. A study is currently underway by the author to compare experimental data with recent parameterizations of the SW potential \cite{7} and MD simulations of the threshold displacement energies for silicon \cite{12}. This study relies on the algorithms presented in this paper to produce timely and accurate results.

In addition to point defects, plane defects can be simulated by displacing an entire algorithmic volume (as in Figures 3 or 12) for as many planes as desirable. This allows for a faster way to test the transferability of plane defects to other parameterizations of potentials. Either point or plane defects could be implemented as single occurrences, or uniform occurrences at regular intervals. Uniformly spread defects or point defects at the origin would still allow use of all the algorithms presented.

IV. CONCLUSIONS

A series of algorithms has been developed for fast calculation of any distant-dependent property of lattices, including imperfect lattices. The algorithms are adaptable to simulate point or plane defects for fitting or testing transferability of parameters in classical potential formulas, and the speedup achieved allows for relaxation of cut-off parameters. These algorithms can be used in serial or parallel, with the greatest speedup achievable through parallelization. As an example of the power of the new algorithm, the Lennard-Jones lattice constants were determined up to 32 significant figures, extending their known precision, and in some cases correcting published figures.

V. ACKNOWLEDGMENTS

The author would like to express his sincere gratitude to his advisor, Jodi Cooley, for her continuous support of his studies. The author would like to thank Randy Scalise and John Fattaruso for the inspiration and assistance with this project. The author is grateful to the SMU Center for Scientific Computation and Amit Kumar for the incredible amount of support and CPU time that made this all possible. The author would also like to thank the SuperCDMS (Cryogenic Dark Matter Search) collaboration and its members for their feedback.

[1] J. E. Jones and A. E. Ingham, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 107, 636 (1925).
[2] R. A. Buckingham, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 168, 264 (1938).
[3] F. H. Stillinger and T. A. Weber, Phys. Rev. B 31, 5262 (1985).
[4] Z. Jian, Z. Kaiming, and X. Xide, Phys. Rev. B 41, 12915 (1990).
[5] M. Timonova, B.-J. Lee, and B. J. Thijssse, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 255, 195 (2007), computer Simulation of Radiation Effects in Solids.
[6] R. Vink, G. Barkema, W. van der Weg, and N. Mousseau, Journal of Non-Crystalline Solids 282, 248 (2001).
[7] L. Pizzagalli, J. Godet, J. Günol, S. Brochard, E. Holmström, K. Nordlund, and T. Albaret, Journal of Physics: Condensed Matter 25, 055801 (2013).
[8] W. Kahan, Commun. ACM 8, 40 (1965).
[9] P. Schwerdtfeger, N. Gaston, R. P. Krawczyk, R. Tonner, and G. E. Moyano, Phys. Rev. B 73, 064112 (2006).
[10] D. C. Wallace and J. L. Patrick, Phys. Rev. 137, A152 (1965).
[11] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc Web page," http://www.mcs.anl.gov/petsc (2016).
[12] The IEEE-754 floating point standard defines quadruple-precision, 128-bit floats. These are implemented differently by various compilers as long-double, _float128, PetscReal, and others.
[13] E. Lusk, N. Doss, and A. Skjellum, Parallel Computing 22, 789 (1996).
[14] J. Frenkel, Zeitschrift für Physik 35, 652 (1926).
[15] E. Holmström, A. Kuronen, and K. Nordlund, Phys. Rev. B 78, 045202 (2008).
| L_p | SC        | BCC       | FCC       | HCP       | DIA       |
|-----|-----------|-----------|-----------|-----------|-----------|
| L_p | 16.5328   | 22.6387   | 25.3382   | 25.3908   | 10.2384   |
|     | 10.375243 | 14.967518 | 14.459321 | 16.964365 | 6.31176058 |
| L_p | 8.40192937 | 12.256876 | 14.359702 | 14.459727 | 5.11677118 |
| L_p | 7.46705778 | 11.042434 | 13.359387 | 13.360476 | 4.39476025 |
| L_p | 6.94580792 | 10.351972 | 12.801972 | 12.802812 | 3.59139479 |
| L_p | 6.26859188 | 9.89458656 | 12.49576021 | 12.4933125 | 4.19037215 |
|     | 6.42611912 | 9.56440966 | 12.31124566 | 12.31189633 | 4.11102399 |
|     | 6.29229449 | 9.31326237 | 12.20092035 | 12.20147099 | 4.06548798 |
|     | 6.20149045 | 9.11813268 | 12.13809165 | 12.13229769 | 3.98094721 |
|     | 6.14095958 | 8.95180738 | 12.08776231 | 12.08804555 | 3.92325118 |
|     | 6.09814152 | 8.61767028 | 12.05991944 | 12.05922855 | 3.86396088 |
|     | 6.05876429 | 8.70294505 | 12.04020405 | 12.04197144 | 3.80405227 |
|     | 6.04826346 | 8.60652404 | 12.02735448 | 12.02749149 | 3.74587078 |
|     | 6.03392361 | 8.52353250 | 12.01889436 | 12.01889771 | 3.68760856 |
|     | 6.02388170 | 8.42553168 | 12.01299380 | 12.01306021 | 3.62957557 |
|     | 6.01628556 | 8.39135079 | 12.00901644 | 12.00906224 | 3.57140891 |
|     | 6.01186280 | 8.38366045 | 12.00628640 | 12.00639151 | 3.51324375 |
|     | 6.00857675 | 8.32930570 | 12.00440451 | 12.00445108 | 3.45507956 |
|     | 6.00596526 | 8.25367218 | 12.00368178 | 12.00368178 | 3.39691547 |
|     | 6.00417024 | 8.21852438 | 12.00211499 | 12.00216086 | 3.33874139 |
|     | 6.00294581 | 8.19015547 | 12.00151092 | 12.00155683 | 3.28055543 |
|     | 6.00205205 | 8.16454553 | 12.00102270 | 12.00106097 | 3.22236953 |
|     | 6.00103473 | 8.12345983 | 12.00052660 | 12.00056285 | 3.16418363 |
|     | 6.00071411 | 8.10692107 | 12.00037127 | 12.00037232 | 3.10599774 |
|     | 6.00005192 | 8.09252928 | 12.00021871 | 12.00023457 | 3.04781185 |
|     | 6.00003677 | 8.08015749 | 12.00018470 | 12.00018522 | 3.00000000 |

**TABLE IV.** The Lennard-Jones lattice coefficients $L_p$ in the SC, BCC, FCC, HCP, and DIA lattices.