Computer Physics Communications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computer Physics Communications

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

A fast, parallel algorithm for distant-dependent calculation of crystal properties

Matthew Stein

Southern Methodist University, 3215 Daniel Ave., Dallas, TX, 75205, USA

ARTICLE INFO

Article history:
Received 10 April 2017
Received in revised form 26 July 2017
Accepted 1 September 2017
Available online xxxx

Keywords:
Parallel algorithms
Crystal potential energy
Lattice constants
Computational approaches
Classical potentials

ABSTRACT

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.

1. 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 [1] is simple and widely used for its computational speed, but much more accurate models exist. The Buckingham potential [2] expanded on the Lennard-Jones potential, replacing the Pauli repulsive term with an exponential function but at computational cost. The Stillinger–Weber potential [3] (hereafter SW potential) was proposed as a further improvement, now taking into account not just 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 [4–7], 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.

2. 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.

E-mail address: mstein@smu.edu.

http://dx.doi.org/10.1016/j.cpc.2017.09.001
0010-4655/

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 3.

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 [8]. 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 applied.

2.1. An illustrative example

The Lennard-Jones potential [1] 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 Eq. (1) between all pairs of atoms. The constant parameters σ and ϵ 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.

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

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

$$L_p \equiv \sum_{j=1}^{\infty} \left(\frac{1}{d_j} \right)^{p}. \tag{2}$$

It is seen that Eq. (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 [9], 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 [9,10].

To achieve a useful value of the lattice constants L_p in Eq. (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 [9], 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 [10], 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 (PETSc) [11] was used to implement 128-bit floats to push the precision to 32 decimal digits.

2.2. Brute force method

Consider a SC lattice whose side length is D, and whose unit cell has a side length of 1 in arbitrary units. To calculate a distance-dependent function $f(r_{ij})$ over all lattice sites (Eq. (2)), one can set up three nested **for**-loops to cover a 3-dimensional grid. Each integer value of the respective loop variables (X, Y, Z) represents the coordinates of a particular atom, and sweeping from $-(D/2)$ to $(D/2)$ in all three loops covers all $(D+1)^3$ atoms in the cube.

The distance d_j from the origin to any other atom j is, of course, $\sqrt{X^2+Y^2+Z^2}$ so the program structure then is:

Algorithm 1 Brute Force Method

```
L_p = 0
for X ← −(D/2) to (D/2) do
    for Y ← −(D/2) to (D/2) do
        for Z ← −(D/2) to (D/2) do
            if X = 0 and Y = 0 and Z = 0 then
                Next
            else
                L_p += 1/(X^2+Y^2+Z^2)^{p/2}
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 1, it is worth finding how many terms will be necessary for this sum to converge.

2.3. The convergent series

Depending on implementation of 128-bit floats [12], these variables yield $\sim$ 32 decimal digits for each term. Finding where Eq. (2) converges then requires additional terms to be equal to or less than 10^{-33} (in arbitrary units). Finding the coordinates of where $L_{p_j} = 10^{-33}$ yields little benefit, however, as that is only the value of one such term, and there may be many such terms at that distance.

For example, say $L_{p_j} = \frac{1}{(X_j^2+Y_j^2+Z_j^2)^{p/2}} = 10^{-33}$ for atom j at (X_j, Y_j, Z_j), and say $Y_j = Z_j = 0$ for simplicity. In the brute force method described above, the algorithm will still be computing approximately R^2 more terms for the face at $X_j = R$. Moreover, there will be six such faces to add to the total sum. Higher distances decrease the value of each L_{p_j} term, but there are more terms to the total sum at some fixed R, slowing down the convergence of the series with increasing distance (Fig. 1). One can calculate the total amount added to L_p from adding one layer at a fixed R distance, showing the slowness of convergence. For L_6, the total value added from one layer at distance R goes as $1/R^4$ (Eq. (3)). This is determined by integrating Eq. (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 Eq. (2), this result is useful for determining how many terms are required for convergence to a particular precision.

$$Sum_{p,face@R} \propto \frac{1}{R^{(p-2)}}$$

$$Sum_{6,face@R} = 6 \times \frac{2 + 15\sqrt{2}\mathrm{ArcCot}\sqrt{2}}{12R^4} \approx \frac{7.52815}{R^4}. \tag{3}$$

The convergence of Eq. (2) is much faster for higher values of p (Fig. 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.

2.4. Finding speedup

2.4.1. 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