Hardening Mechanisms of Silicon Nanospheres: A Molecular Dynamics Study

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Lucas Michael Hale

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

William W. Gerberich, Roberto Ballarini

May 2011

Acknowledgements

I'd like to thank Dong-Bo Zhang and Traian Dumitrica for providing the DFTB calculations used in Chapter 4. Additionally, I want to acknowledge Xiaowang Zhou, Jonathan Zimmerman and Neville Moody at Sandia National Laboratories in California for supporting my research.

Funding for this work was partially supported in part by National Science Foundation grants NSF_CMMI 0800896 and CMMI-1000415. Additional support was obtained by the Air Force through an AOARD-08-4131 program dedicated to understanding plasticity and fracture in hard materials and the Abu Dhabi-Minnesota Institute for Research Excellence (ADMIRE); a partnership between the Petroleum Institute (PI) of Abu Dhabi and the Department of Chemical Engineering and Materials Science of the University of Minnesota. Work was also supported by Sandia, Livermore. Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DEAC04-94AL85000.

Abstract

Much work has been done studying the compression of nanostructures of silicon as the measured properties can be related to structures present in MEMS and NEMS devices. In particular, spherical silicon nanoparticles are found to be much harder than bulk silicon during compression. Here, large scale molecular dynamics simulations are presented that investigate the yielding and hardening mechanisms of nanospheres. The resulting yield behavior is shown to vary with changes in temperature, sphere size, atomistic potential, and crystallographic orientation with respect to the loading direction. With the Tersoff potential, a strong temperature dependence is observed as hardness values near 0 K are much greater than 300 K values. β-Sn forms during [100] crystallographic compressions which results in a slight hardening above 40 % strain. The Stillinger-Weber allowed for dislocation interactions to be studied in spheres comprised of up to one million atoms. Direct comparisons of the simulated results are made to experimental results indicating that the displacement excursions and low strain hardening behavior can be explained with dislocation activity. Further simulations investigated interactions affecting dislocations that might influence the properties of silicon nanostructures. The nature of dislocation-dislocation, dislocation-applied shear strain, and dislocation-free surface interactions are shown to be consistent with what is predicted by elementary dislocation theory. Presence of an oxide results in a more complex interaction as both the interface and the lattice strain associated with the oxide affect the dislocations. Depending on the geometry of the system, this oxide interaction may be repulsive resulting in dislocations becoming trapped in the system allowing for substantial hardening.

Table of Contents

List of Tables

List of Figures

Chapter One: Introduction to Silicon Nanostructures

1.1 Motivation

Over the last few decades, most fields of engineering have had a fixation on one word: Nano. The fixation on the nano-scale can be considered nothing more than a continuation of the miniaturization of technology that had already seen the entrance into and the mastery of the micro-scale. But in other ways, the nano-scale introduces new challenges and interesting discoveries to be made. Many physical properties no longer follow the trends exhibited at larger scales and as a result a material's behavior, mechanical or otherwise, deviates from the expected. This size scale effect is most often attributed to the increased influence that surfaces have on material behavior and on the system size beginning to approach the same scale as atoms and molecules.

Inevitably, the study of particles and structures on the nano-scale leads to the question of practicality: can devices be successfully designed to incorporate such small features? Inherent to answering this is determining if the devices are mechanically robust and will operate with sufficiently high levels of reliability. Designing and implementing devices at this size requires a proper understanding of not only how these behaviors deviate from the expected larger scale behaviors, but also why. To better describe the importance of this, a handful of potential applications and the questions related to their mechanical reliability are described here.

First, consider a silicon MEMS or NEMS device that contains free standing features of many sizes, including some at the nano-scale. The most critical failure

mechanism of MEMS devices is adhesion wear due to contact between different constituent components. This failure mechanism results from repeated contact and rubbing between surfaces causing damage to nano-scale surface asperities [1]. An understanding of this phenomenon requires knowledge of how these asperities are plastically damaged.

Next, consider a film of nano-grained silicon that is subjected to a surface loading. How would the nano-grained film behave differently than a single crystal film? Would it offer more or less protection to the material that it coats? Would the yielding that occurs within the grains differ from bulk silicon? If nanoparticles of comparable size to the grains showed an increase in hardness behavior, would also the film? And what if the film was synthesized from nanoparticles?

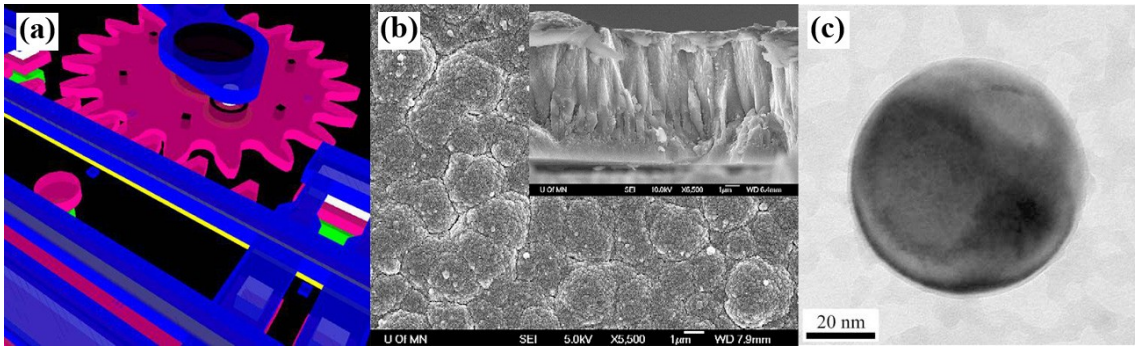

Figure 1.1: SEM images of examples where the study of the mechanical nature of nanostructures is useful. (a) Image of a silicon MEMS device featuring a gear subjected to repeated contact [2]. (b) Image of the surface of a SiC film created by depositing nanoparticles [3]. (c) Image showing a silicon nanoparticle between 40 and 50 nm in diameter [4]. This particle was used for experimental testing of these nanoparticles. It is also of comparable size to particles used in polishing slurries.

Finally, think about a polishing slurry that contains particles of nearly uniform size suspended within a solution. This perhaps offers the simplest example of particles

subjected to mechanical loads, as the particles in the slurry are used to rub against the surface of the material that is being polished. The smaller the particles used in the slurry, the finer the damage that they cause, thus a smoother finish. But are the smaller particles capable of retaining their shape and size during such a process, or will they yield and/or fracture?

In order to investigate the mechanical behaviors associated with these questions, a host of techniques have been developed to create, study, and model sub-micron particles. With regards to modeling, upon reaching the nano-scale, the classical continuum based modeling paradigms used to design larger-scale objects are often no longer applicable as atomic scale properties become more relevant. Instead, atomistic based techniques are required, such as molecular dynamics and Monte Carlo simulations. What is perhaps most interesting about these atomistic techniques is that as the computational efficiency and power advances, the largest dimensions that can be readily simulated are approaching the scale that can be measured experimentally. This allows for increasingly more opportunities to directly compare these types of simulations to the experimental results.

The focus of this thesis is the molecular dynamics simulations related to the study of the mechanical compression of silicon nanoparticles. It is hoped that the simulations will reveal the specific mechanisms for yielding that occur and so far have only been inferred experimentally. This would not only help identify possible mechanisms of mechanical response, but also allow new and more accurate theoretical models to be developed. In addition, as no atomistic simulation is completely accurate, comparing these simulations to experimental results will offer a better understanding of

which assumptions made in the simulations result in the most realistic response of real material response.

1.2 Experimental Background

Because the molecular dynamics simulations that comprise this thesis are designed to best mimic experiments, it is necessary to understand how the baseline experiments were performed and the results that they produced. A short background is presented here to describe nanoindentation, the experimental technique most often used, and to discuss the experimental results obtained from bulk silicon and silicon nanostructures.

1.2.1 Nanoindentation Basics

The experimental works that focused on compression of nanoparticles and structures are based upon the use of nanoindentation apparatus and the theories and principles developed for their use. Instruments commonly referred to as nanoindenters use hard tipped probes to measure the mechanical resistance of materials to contact deformation. The shape of the indenter tips are well defined and are often spherical, conical, or pyramidal. Unlike macro-sized indenters that measure material hardness values, there is no standardized tip size across different manufacturers, although sizes are typically in the range of the spherical tip having a radius of curvature of 1 μm. This, along with the sharp features of the other tip geometries, allows for the probing of nano-sized regions.

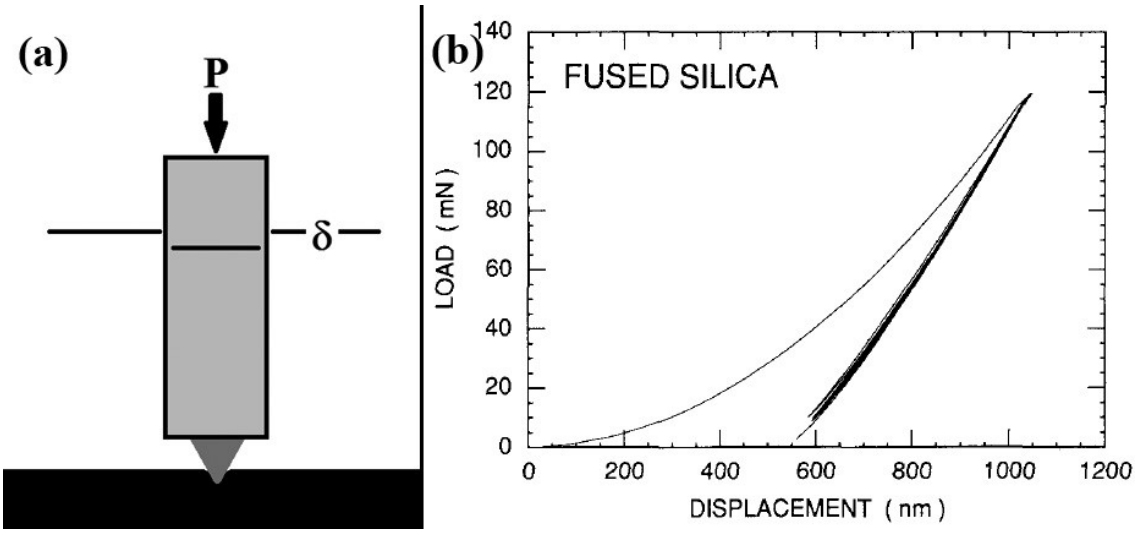

Figure 1.2: (a) A diagram showing a simplified representation of nanoindentation. The tip is brought into contact with the surface and is then subjected to an applied load, P, driving the tip into the material. The displacement is determined by comparing the position of the tip at the surface relative to its position with the applied force. (b) A typical load vs. displacement curve obtained during nanoindentation [5]. The fused silica was indented up to a load of 120 mN resulting in residual displacement after unloading showing that elastic and plastic behavior occurred during loading. After reaching the maximum force, the load was cycled. All unloading and cycling are nearly on top of each other suggesting that the unloading and reloading is nearly elastic.

During testing, nanoindentation devices measure two values: the applied force and the position of the tip. The displacement into the surface can then be calculated by identifying the location where the tip first comes into contact with the surface. Indentation experiments are typically load controlled, meaning that load vs. displacement plots are obtained by contacting the indenter tip on the surface of the material being tested, then ramping up the applied load while measuring the tip's position. This is done as the instruments are more apt at controlling an applied load than they are for applying a specific displacement. A simple schematic of the indenter tip and an example of a load vs. displacement curve are shown in Figure 1.2.

In 1992, Oliver and Pharr [5] published a standardized technique for calculating the elastic modulus and hardness of indented materials. Their method has been readily accepted and used throughout the indentation community. Within their work, they showed that the complex expressions for the stiffness, *S*, with an axisymetric indenter can be reduced to the expression

$$S = 2E_r a = \frac{2}{\sqrt{\pi}} E_r \sqrt{A} \tag{1.1}$$

where E_r is the effective modulus of the tip and surface, A is the contact area and a is the radius of the contact area. By observing that during unloading most materials behave elastically, the slope of the unloading curve can be used to measure the stiffness of the material. This allows for either that material's elastic properties or the contact area to be determined, assuming that one is known. The hardness of the material, H, was also defined as the applied load, P, divided by the contact area.

$$H = \frac{P}{A} = \frac{P}{\pi a^2} \tag{1.2}$$

The Oliver and Pharr method [5] relies on the knowledge of the contact area between the indenter tip and the surface. Although very important in acquiring accurate values, the contact area cannot be directly measured. Also, many factors can influence it, including tip and surface imperfections, yield point values and material anisotropy. Many methods have been developed to estimate the contact area. The simplest are elasticity [6] and geometric [7] based models that assume values for the contact area under ideal conditions. More complex methods involve obtaining a surface profile of the specific tips used allowing for all of the tip's defects to be accounted for in a contact area function for the tip used [8].

1.2.2 Pressure induced phases of silicon

One of the more interesting aspects of the mechanical properties of silicon is the fact that it can undergo numerous pressure induced phase transformations at relatively low pressures. These phase transitions have been extensively studied with diamond anvil pressure tests [9-13]. The fact that the stresses due to a nanoindenter are large enough to at least locally induce some of these phase transformations has resulted in numerous nanoindentation studies on bulk silicon. Exploring these studies and the possible phase transformations offers a good starting point because the experimental and computational techniques are similar to what is used to study nanostructures. It also allows for a direct comparison between the behavior of the nanostructures and that of the bulk material.

The most thermodynamically stable structure is the diamond cubic structure, often referred to as either DC or Si I. In this structure, each atom covalently bonds to four neighbors to form a tetrahedra. This covalent structure gives silicon some unique and interesting properties. This structure is relatively brittle and elastically anisotropic. In addition, due to the relatively loose atomic packing of this structure, it is capable of transforming into a number of pressure induced phases. The two of the high pressure phases that are observed during the simulations presented here are also described.

The β-Sn (Si II) phase is the first of the high pressure phases that silicon is known to form during hydrostatic compression [11]. Its unit cell is shown in Figure 1.3(b). This phase can be viewed as a tetragonal distortion of the DC structure through a reduction in the lattice parameter in one of the three cubic directions. The resulting

structure has all atoms with six nearest neighbors and is metallically conductive. β-Sn Si is not stable at ambient pressures and at room temperature will revert to a variety of four coordinated metastable polymorphs [9, 10, 14-16] and amorphous silicon. These phases can all be seen as structures that form due to a distortion in the tetrahedral bonding found in the DC structure. The amorphous silicon phase is best described as being locally bonded similarly to the other four coordinated structures but lacking an overall crystalline periodicity.

Higher pressures result in a series of other structures being formed [13]. The pressures necessary to form these phases are not considered in this thesis, and therefore they will not be described here.

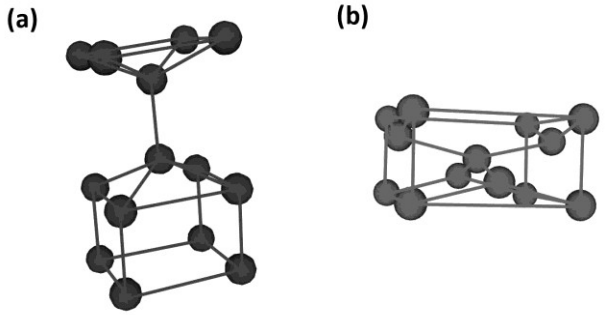

Figure 1.3: Representations showing the bonding nature of the (a) BCT5 and (b) β-Sn structures of silicon.

In addition to the well known and experimentally observed phase transformations, another phase referred to as BCT5 is seen during atomistic simulations. The BCT5 phase was first proposed by Boyer, et al. in 1991 [17] and its unit cell and bonding is shown in Figure 1.3(a). Initially identified using a Stillinger-Weber potential [18] and first principle calculations, BCT5 is a body-centered-tetragonal structure where

every atom has a coordination number of 5. The Stillinger-Weber results showed BCT5 to be stable, while the first principle pseudo potential had it being metastable at all pressures. Interestingly, like the β-Sn structure, BCT5 can also be obtained through a distortion of the DC lattice [19]. One simple way of looking at this distortion is taking a glide set of [111] planes in the DC structure. Both of the planes in this set consist of a 2-D hexagonal structure and are positioned with respect to each other such that each atom is neighboring three atoms in the other plane. BCT5 type bonding is then achieved by deforming both planes to square lattices resulting in an additional atomic bond for each atom (Figure 1.4). Simulations using the Tersoff atomistic potential [20, 21] have also shown the BCT5 phase to form during loadings [19, 22-25].

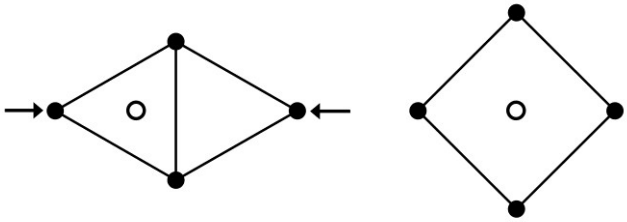

Figure 1.4: A diagram allowing for a visualization of how diamond cubic silicon can transform to BCT5 upon loading. (a) The relative positions of atoms within two neighboring [111] planes in silicon of the glide set. Each atom is bound to 3 atoms in the other [111] plane shown, and one atom directly above/below in another plane. (b) The planes are deformed from the hexagonal packing to the square structure resulting in each atom gaining a neighbor.

1.2.3 Nanoindentation of bulk silicon

It is generally accepted that indenting bulk silicon induces phase transformations. There have been numerous experimental indentation studies of silicon

[26-40] using nanoindenters and their predecessor techniques. These tests repeatedly revealed three key pieces of evidence suggesting that the phase transformation is occurring. First, the conductivity changes from that of a semiconductor to that of a metal upon loading and reverts back to a semiconductor upon unloading. This supports the formation of β-Sn due to it being a metallic phase. Second, the load vs. displacement curve displays unique characteristics. The unloading curves often feature what are referred to as "elbows" or "pop-out" where either the slope of the curve or the curve itself is discontinuous indicating that there is a recovery mechanism occurring (Figure 1.5). It is clear from Figure 1.5(b) that a phase transformation has occurred during unloading as the curve during reloading is distinctly different from the unloading curve. The final evidence for this transformation is the presence of amorphous Si and the Si III/XII phases after unloading.

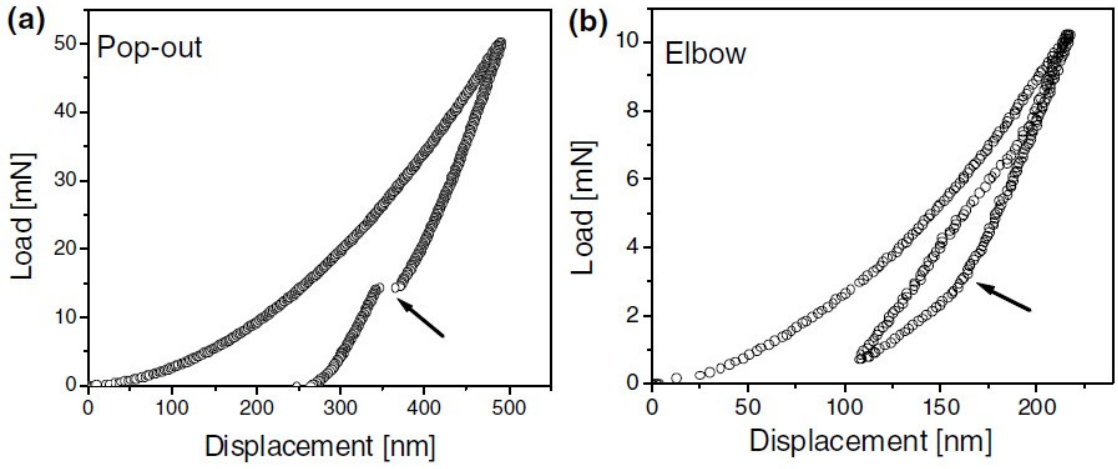

Figure 1.5: Examples of the unloading behaviors seen during the nanoindentation of silicon [28]. (a) Shows the pop-out behavior consistent with a fast unloading rate resulting in the formation of amorphous Si. (b) Shows an elbow typical of a slow unloading allowing for the formation of various 4 coordinated structures.

Much of the work on Si nanoindentation has investigated the loading conditions and the resulting phases obtained after unloading. The most notable result is that slow unloading rates increase the likelihood for Si III and Si XII to form and correspond to the elbow behavior, whereas fast unloading rates leave behind amorphous silicon after a pop-out event [27, 28].

Molecular dynamics simulations of the indentations using the Tersoff potential support the experimental findings [19, 22-25, 41-43]. Upon indentation, the atoms below the tip compress and increasingly gain more nearest neighbors. Based on this coordination number change, not only is the β-Sn phase identified in this compressed region, but also the BCT5 phase. A cross sectional example is seen in Figure 1.6. Pressure induced deformation pathways have been identified allowing for these phase transformations to occur without any bond breaking allowing the coordination number to smoothly increase and the phases to gradually form [19]. Unloading and annealing shows the compressed region relaxing and reverting back to 4 coordinated bonds.

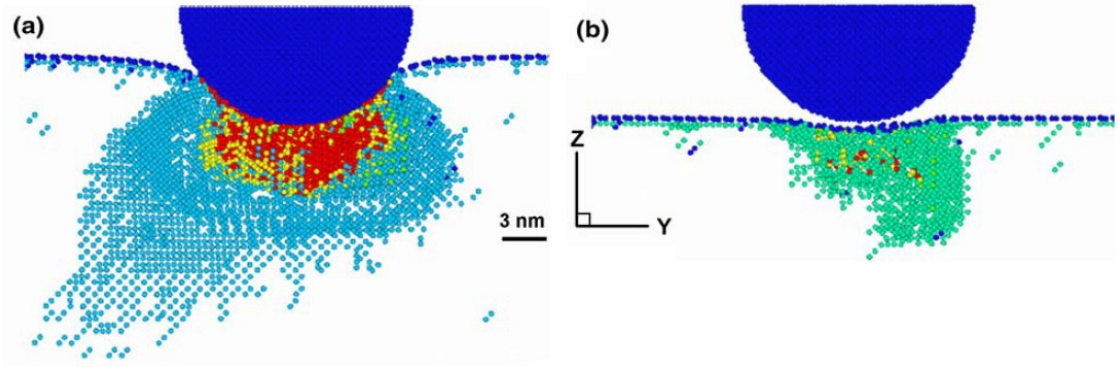

Figure 1.6: Cross section of a molecular dynamics simulation using the Tersoff potential with atoms colored according to their coordination [24]. (a) The region just below the indentation shows high pressure red atoms mixed with yellow 6 coordinated β-Sn. 5 coordinated atoms believed to be BCT5 are then the light blue atoms radiating out below the red and the yellow atoms. (b) While the majority of atoms revert back to a 4 coordinated structure upon unloading, some residual damage is still present.

1.2.4 Compressing Si nanostructures

While the yielding mechanisms occurring during nanoindentation of bulk silicon have been well investigated, the same cannot be said for nanostructures. A much more rich and complex yielding system appears to be involved that includes not only the pressure induced changes, but also dislocations. For the nanostructures, the resulting deformation mechanisms are influenced by their size and shape as these dictate the stress fields present within and also increase the interaction and effects due to the surface.

Experimental compression of nanoparticles has revealed much evidence to support dislocation yielding to be the primary yielding mechanism [4, 7]. Upon compression, numerous excursion events of equal magnitude are observed in the load vs. displacement plot. Such load drops are often associated with dislocation activity. A

marked increase in the hardness is also observed for repeated loadings suggesting that whatever permanent deformation has occurred interferes with subsequent yielding. As dislocation interaction is a well known hardening mechanism, it is plausible that this hardening is the result of an increase in the number of dislocations that are present within the confined volumes of the particles. Furthermore, no clear unloading elbows or kinks characteristic of a reverse phase transformation were reported.

These works were followed by others that involved using a TEM and an *in situ* indenter to directly observe the yielding behavior [44-49]. For the nanoparticles, regions of plastic damage were observed near the contact, but the nature of this damage was never addressed as no single dislocation loops were ever separated and no diffraction peaks indicative of the β-Sn phase were observed. The load vs. displacement found for one of these spheres is shown along with before and after TEM images in Figure 1.7.

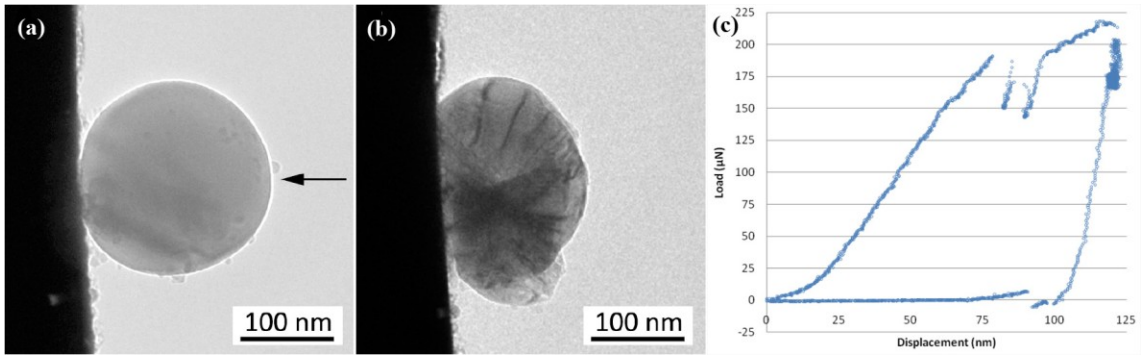

Figure 1.7: (a)+(b) TEM images of a silicon nanoparticle before and after compression with an in situ nanoindenter. (c) The load vs. displacement for that sphere compression [48].

Direct evidence of dislocations within silicon have been observed within other nanostructures. Minor, et al. [50] created TEM transparent wedges of silicon and

observed dislocation loops after indenting as opposed to the phase transformation. This was attributed to the geometry resulting in shear stress concentrations not observed during bulk indentation. Compression of nanopillars have indicated a size-dependent brittle-ductile transformation where silicon pillars behaved plastically below a certain pillar diameter [51]. *In situ* observations of pillars have also revealed what appears to be dislocation activity travelling from the indenter down inside the pillars [52].

Simulations of silicon nanostructures have reported a wide variety of different yielding. The most widely studied structural formation has been silicon nanowires pulled in tension [53-57]. These revealed that extensive dislocation plasticity is possible resulting in a ductile fracture of the wires.

For compression, a wide range of possible yielding behaviors have been reported. One of the first papers on experimental compression of silicon nanoparticles also included work related to a silicon nanosphere compressed with MD using the MEAM interatomic potential [7]. From this, only amorphous damage was observed, most likely due to the small simulation size and time resulting in extremely fast loading. Only amorphous damage was also reported by Fang, et al. [58] for MD simulations of compressed nanocubes using the Stillinger-Weber potential. However, it should be noted that the total strain and deformation from these simulations was relatively small and focused only at the initial yielding behavior.

Another series of MD simulations by Valentini and Dumitrica [59, 60] related to nanospheres of silicon revealed that the Tersoff potential allows for extensive regions of β-Sn to form during compressive loads (either due to uniaxial compression or impact with a substrate). For the uniaxial compression, this phase transformation was seen to

accompany a hardening behavior at high strains which increased with repeated loadings at 0 K [60]. Figure 1.8 shows the sphere containing a region of β-Sn that remained after unloading. Annealing this sphere resulted in the β-Sn reverting back to a 4 coordinated structure. With the high velocity particle impact simulations, the spheres and the substrate were passivated with hydrogen to eliminate free bonds at the surface. Adhesion was seen to occur only when the velocity was high enough to cause plastic deformation within the sphere [59].

Zhang et al. [61] also performed MD simulations on silicon nanoparticles. Comparing results from the Stillinger-Weber and Tersoff potentials, they noted that the Stillinger-Weber potential shows dislocations as the predominant yielding mechanism while the Tersoff potential favored the β-Sn phase transformation. The applied load and average compressive stress within the spheres was shown to be comparable for the two potentials prior to the first yielding events. After this yielding, the two potentials deviated with the Tersoff reaching considerably larger applied loads. The maximum hardness reported for a 40 nm sphere were 5.58 GPa for the Stillinger-Weber potential and 30.9 GPa for the Tersoff potential. Because of this, they concluded that the Tersoff potential was more consistent with the experimental results.

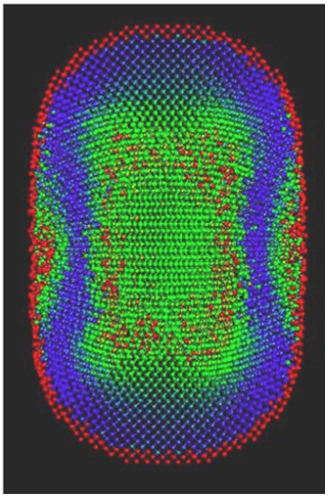

Figure 1.8: A cross sectional image from an MD simulation of a silicon nanosphere that had been compressed. The colors denote the relative potential energy of the atoms. Note that the center of the sphere has a set potential energy and structure indicating that it has transformed into β-Sn [60].

Finally, a recent paper by Yang, et al. [57] investigated silicon nanowires in compression. This gives direct information into the nanopillar compression. Using the Stillinger-Weber potential, they observed dislocation activity nucleating on the {110} type planes. As this type of dislocation activity is not known to occur for silicon, it was attributed to being indicative of possible cleavage.

Chapter Two: Simulation Procedure

2.1 Molecular Dynamics Background

Molecular dynamics, MD, is an atomistic simulation technique in which the positions and velocities of all the atoms are updated with time. At its most basic and simplest form, there are only two steps involved in MD: using potential functions to calculate the forces acting on the atoms based on the atom's positions and numerically integrating the positions and velocities of the atoms based on the forces. While conceptually simple, MD quickly becomes complex to implement as both of these steps need to be calculated for every atom at every recursive instance of time. This makes efficient and powerful computing techniques and equipment a necessity for simulating a large number of atoms for an extended period of time.

2.1.1 Atomistic potentials

The forces acting on the atoms can be divided into the internal atom-atom interactions and the externally applied system constraints. Typically, these forces and interactions are calculated using mathematical models of the potential energy associated with the position of the atoms with respect to each other and with respect to the total system. These models of the potential energy are commonly referred as potentials. The position dependent force interactions are then found by differentiating the potentials with respect to position. The resulting force equations are referred to as force fields.

There are numerous interatomic potentials that have been developed for a wide variety of materials. These potentials vary from the incredibly simple that only contain an attractive term and a repulsive term for the interaction between two atoms, such as the Lennard-Jones potential [62, 63], to the very complex involving terms that depend on the position, number and composition of all neighboring atoms.

Conflicting concepts are associated with classical interatomic potentials for MD. Ideally one would prefer to use the simplest functional form that can represent the correct material behavior while also being computationally efficient. However, experience has shown that a particular potential faithfully reproduces only certain materials under certain conditions. Describing the complete behavior of any one or more materials inherently requires either a more complex potential or multiple potentials.

In addition, classical functions and descriptions of atomic interactions are chosen as the size scale allows for the electronic behaviors to be glossed over and averaged out without knowing the specifics. However, the bonding behavior that the potential functions model are inherently dependent on the electronic properties of the atoms. Changes in nearest neighbor conditions can change the dominant bonding type associated with an atom and result in the chosen potential being unsuitable for the new conditions. This greatly detracts from the possibility of a universal potential ever being created. But if there was a way to create a series of rules that allow for a conditional potential, or combination of potentials, to address the different possible bonding types, the range of effective modeling would be increased.

The potential that is typically used as an example when describing MD is the classic and simple Lennard-Jones potential [62, 63]. This potential is referred to as a two-body potential because it only depends on the distance between two atoms. The typical form used for this potential is

$$E = 4\kappa\left[\left(\frac{\rho}{r}\right)^{12} - \left(\frac{\rho}{r}\right)^{6}\right] \tag{2.1}$$

Where r is the interatomic distance and κ and ρ are scaling constants with units of energy and distance respectively. This simple potential is easily differentiable resulting in a force field equation of:

$$F = -24\kappa\left[2\frac{\rho^{12}}{r^{13}} - \frac{\rho^{6}}{r^{7}}\right] \tag{2.2}$$

The first term in the brackets is repulsive, while the second term is attractive. As the power of the repulsive term is greater than the attractive term, it has little effect at large r and becomes increasingly predominant as r decreases. Overall, the function mimics atomic interactions by attracting atoms that are far apart, and then as the atoms get closer, there is an equilibrium point at $r = 2^{1/6}\rho$ below which the function becomes repulsive preventing the atoms from overlapping each other. At the equilibrium point, the potential energy between the two atoms is $-\kappa$ less than what it would be if the atoms were not interacting. Surrounding the equilibrium point is a potential well which can trap atoms and prevent them from separating resulting in an atomic bond.

While the Lennard-Jones potential allows for an easy understanding of how potentials work, it is too simple to accurately model silicon as the diamond cubic structure has angle dependent covalent bonds. Therefore, other potentials have been

developed that model the interaction with multi-body potentials. Two of the most well known and used potentials are the Stillinger-Weber potential and the Tersoff potential, both of which are used for this research.

2.1.2 External constraints

The forces acting on the atoms can also be modified with external constraints. The use of these constraints depends on the system that is meant to be simulated. As such, they will be addressed later with the simulation procedures.

2.1.3 Time integration

As mentioned before, the forces calculated from the potentials are used to update the position and velocity of the atoms with respect to time. This is accomplished by relating the mass of and the forces on the atoms to their acceleration. The accelerations are then numerically integrated to determine velocity and position. While a number of different methods for this integration exist, the most commonly used method with molecular dynamics is the Velocity Verlet algorithm [64]. This algorithm first uses the position, x, velocity, v, and acceleration, a, at time, t, to calculate an updated position that is Δt later in time

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \tfrac{1}{2} a(t)\Delta t^2 \qquad (2.3a)$$

Then calculates the forces and acceleration at the new step using the potential functions before updating the velocity with

$$v(t+\Delta t) = v(t) + \tfrac{1}{2}\left(a(t) + a(t+\Delta t)\right)\Delta t \qquad (2.3b)$$

This method is ideal for its use with MD as it is simple to calculate, it conserves the total energy of the system and it has a small error under the right conditions.

2.2 Silicon Potentials

Having strongly covalent bonds, the crystal structure of silicon at room temperature and atmospheric pressure is diamond-cubic (DC). This structure, which consists of the silicon atoms having tetrahedral bonding, cannot be adequately modeled solely with a pair style potential as pair styles favor closed-packed and symmetric structures.

2.2.1 Stillinger-Weber potential

One of the first atomic potentials that adequately modeled the diamond-cubic structure of silicon was the Stillinger-Weber (SW) potential [18]. This potential is of the form

$$\Phi = \sum_i \sum_{j>i} v_2\left(r_{ij}\right) + \sum_i \sum_{j\neq i} \sum_{k>j} v_3\left(r_{ij}, r_{ik}, \theta_{ijk}\right) \tag{2.4a}$$

$$v_2\left(r_{ij}\right) = A_{ij}\kappa_{ij}\left[B_{ij}\left(\frac{\rho_{ij}}{r_{ij}}\right)^{p_{ij}} - \left(\frac{\rho_{ij}}{r_{ij}}\right)^{q_{ij}}\right]\exp\left(\frac{\rho_{ij}}{r_{ij} - a_{ij}\rho_{ij}}\right) \tag{2.4b}$$

$$v_3\left(r_{ij}, r_{ik}, \theta_{ijk}\right) = \lambda_{ijk}\kappa_{ijk}\left(\cos\theta_{ijk} - \cos\theta_{DC}\right)^2 \exp\left(\frac{\gamma_{ij}\rho_{ij}}{r_{ij} - a_{ij}\rho_{ij}}\right)\exp\left(\frac{\gamma_{ik}\rho_{ik}}{r_{ik} - a_{ik}\rho_{ik}}\right) \tag{2.4c}$$

In these expressions, r_{ij} and r_{ik} are the radial distances between two atoms and θ_{ijk} is the angle between the triplet of atoms bonded *j*-*i*-*k*. All other terms are fitting parameters

with epsilon and sigma scaling the energy and length scales respectively allowing the remainder of the terms to be dimensionless.

The function v_3 is a three-body interaction term that stabilizes the non-closed packed and non-centrosymmetric DC structure. In particular, the three-body term of the SW potential calculates the angle between the three atoms in question and adds an energy that increases as the angle deviates from what is found in perfect diamond cubic, θ_{DC}. As this term explicitly depends on the DC bond angle, it undoubtedly favors that structure over any other. Many of this potential's properties, both positive and negative, is inherently due to this DC favoritism.

Although many new silicon potentials have been developed over the last 25 years since the SW potential was first published, it still finds much use in the simulations of today. This is partially due to the potential's simple form allowing it to be easily implemented, but also modifiable. It also decently simulates many of the properties of bulk silicon. For instance, the parameters used were chosen such that the lattice parameter and cohesive energy of the diamond-cubic structure would match what is experimentally found. In addition, it has been shown to decently replicate the elastic constants even though it was never optimized for those values [65, 66]. Atomistic defects within DC Si are also well modeled as it creates adequate representations of vacancy and self-interstitial [67-69] structures and has one of the better representations of low temperature dislocation behavior out of the more common silicon potentials [70].

The SW potential suffers in its representation of Si structures that do not have bonding similar to DC. Most notably, it is unable to correctly model surfaces or atomic

silicon clusters and the pressure induced transformation to β-Sn is not observed as the first pressure induced transformation [65].

2.2.2 Tersoff potential

First published a year after the SW potential, the Tersoff potential [20, 21, 71] was initially developed to be capable of realistically representing many of the high pressure silicon phases, including β-Sn, and the transformations between them. Mathematically, this potential has a more complex functional form given by

$$\Phi = \tfrac{1}{2}\sum_i \sum_{j\neq i} f_c\left(r_{ij}\right)\left(A_{ij}\exp\left(-\lambda_{ij}^{(1)} r_{ij}\right) - \left(1+\beta^n \zeta_{ij}^{\,n}\right)^{\frac{1}{2n}} B_{ij}\exp\left(-\lambda_{ij}^{(2)} r_{ij}\right)\right) \quad (2.5a)$$

$$f_c(r) = \begin{cases} 1 & : \quad r < R-D \\ \frac{1}{2} - \frac{1}{2}\left(\sin\left(\frac{\pi}{2}\frac{r-R}{D}\right)\right) & : \quad R-D \le r \le R+D \\ 0 & : \quad R+D < r \end{cases} \quad (2.5b)$$

$$\zeta_{ij} = \sum_{k\neq i,j} f_c\left(r_{ik}\right) g\left(\theta_{ijk}\right) \exp\left(\left(\lambda_{ij}^{(3)}\right)^m \left(r_{ij} - r_{ik}\right)^m\right) \quad (2.5c)$$

$$g\left(\theta_{ijk}\right) = \gamma_{ijk}\left(1 + \frac{c^2}{d^2} - \frac{c^2}{d^2 + \left(\cos\left(\theta_{ijk}\right) - \cos\left(\theta_0\right)\right)^2}\right) \quad (2.5d)$$

Once again, all terms except r_{ij}, r_{ik} and θ_{ijk} are fitting parameters. At first glance, this functional form appears to contain only two-body terms, but a closer examination reveals that ζ in the attractive component is dependent on atoms i, j and k. f_c is a cutoff function that quickly but smoothly reduces the energy to zero over a short range.

In its original form, there are two parameter sets developed by Tersoff that are most commonly used [20, 21]. Commonly referred to as either T2 and T3 or T(B) and T(C), they offer different advantages and disadvantages. T(B) is better capable of

representing some of the silicon atomic clusters and bulk silicon surfaces than either SW or T(C). T(C) was created to better match the elastic properties of DC silicon, but at the cost of the cluster and surface representations. As the mechanical properties of the material are more important for the current study than surface structures, T(C) was used exclusively for this work. T(C) has the distinct advantage over SW by allowing for the β-Sn transformation to occur due to an increase in pressure.

2.3 Si-SiO capable potentials

Modeling a system containing both silicon and silicon dioxide with classical potentials offers an interesting problem. While each material by itself has been adequately modeled with a number of different potential forms, there are few successful efforts at modeling the interactions between the two that have been published. The bonding within bulk Si and the bonding within SiO2 are so vastly different that the equations for modeling their interactions are nothing alike. The way that silicon interacts with other silicon atoms in bulk Si is completely different than how it interacts with those same atoms in bulk SiO2.

Simply using an Si potential and an SiO potential and placing the two simulated materials next to each other is no guarantee that the interface would be realistic as many factors come into play. Should all Si atoms be allowed to interact in the same way, or should the ones in the oxide be defined as being different to accommodate the different bonding? If the oxide is handled as a completely different material, how does one specify the Si-Si and Si-O reactions across the interface?

An extensive search through previous sources reveals only three reported methods to remove this ambiguity. The first, by Jiang and Brown [72, 73], attempted to create a composite potential combining SW with the SiO potential by van Buren, Kramer, and van Satten [74, 75] and adding additional conditional components to allow the two to mix and interact. The second method, by Watanabe, et al. [76, 77], offers a modification to the SW potential that allows for SiO structures to correctly form. The third is by using a much more complex and involved force field, such as the Reax force field [78-80]. Although ReaxFF offers the promise of possibly being a more complete and accurate potential, little has been published about the structure and properties resulting from how it simulates silicon. Because of this, only the first two potentials were investigated for the work presented here and thus will be discussed.

2.3.1 Jiang and Brown Potential

The potential created by Jiang and Brown [72, 73] is based upon creating a simple analytical model combining SW Si-Si interactions with the well used Si-O interactions of the potential by van Beest, Kramer and van Santen (BKS) [74, 75]. The Jiang and Brown potential is of the form

$$\Phi = \sum_i e_i(q_i) + \sum_i \sum_{j>i} g_{ij} v_2(r_{ij}) + \sum_i \sum_{j>i} \phi_{BKS}(q_i, q_j, r_{ij}) + \sum_i \sum_{j\neq i} \sum_{k>j} g_{ij} g_{ik} v_3(r_{ij}, r_{ik}, \theta_{ijk}) \tag{2.6a}$$

In this formulation, v_2 and v_3 are respectively the two- and three-body terms of the SW potential, ϕ_{BKS} is the BKS pair potential.

$$\phi_{BKS} = \frac{q_i q_j}{r_{ij}} + A_{ij}\exp\left(-b_{ij} r_{ij}\right) - \frac{c_{ij}}{r_{ij}^6} \tag{2.6b}$$

The charge on every atom is calculated as a function of coordination with the opposite atom type, with a smoothing function used at the cutoff distance to eliminate discontinuous behavior.

$$q_i \equiv \begin{cases} \sum_{j\subset O} q_O H\left(r_{ij}\right) & i \subset Si \\ -\sum_{j\subset Si} q_O H\left(r_{ij}\right) & i \subset O \end{cases} \tag{2.6c}$$

$$H\left(r_{ij}\right) \equiv \begin{cases} 1 & r \le r_o \\ \frac{1}{2} + \frac{1}{2}\cos\left(\pi \dfrac{r - r_o}{r_s - r_o}\right) & r_o \le r \le r_s \\ 0 & r_s \le r \end{cases} \tag{2.6d}$$

The additional terms are charge dependent functions designed to integrate the two different material behaviors given by SW and BKS together. The first, e_i, is an ionization term that adds a penalizing energy to any atom that has a charge corresponding to more than 4 O neighbors for each Si and more than 2 Si neighbors for every O.

$$e_i = \begin{cases} e_o \exp\left[\dfrac{-1}{\left|q_i - q^o\right|}\right] & \left|q_i\right| > \left|q^o\right| \\ 0 & otherwise \end{cases} \tag{2.6e}$$

Lastly, g is a bond softening function that weakens the SW Si-Si interaction as the charges on the incorporated Si atoms increase

$$g_{ij} = \begin{cases} \exp\left[\dfrac{1}{q_s}\right]\exp\left[\dfrac{1}{q_i + q_j - q_s}\right] & q_i + q_j < q_s \\ 0 & q_i + q_j \ge q_s \end{cases} \tag{2.6f}$$

Within this potential, all of the parameters associated with the SW and BKS components were left unchanged. Fitting was then done for all of the new parameters, q_O, r_s, r_o, q^o based upon placing a single O interstitial impurity in a bulk Si material. From this, the structure and stability of larger defect complexes were then examined.

2.3.2 Watanabe, Fujiwara, Noguchi, Hoshino and Ohdomari Potential

This potential, which will be referred to as the Watanabe potential solely for simplicity, is a modification of the SW potential that allows it to also simulate bulk SiO_2 crystal structures [76, 77]. In form, it is identical to the SW potential except that it contains a coordination-based bond softening function, g_{ij}, which acts on the two-body term for Si-O interactions. The bond softening function in the Watanabe potential is given by the following expressions

$$g_{ij} = \begin{cases} g(z_i) & i = O, j = Si \\ g(z_j) & i = Si, j = O \\ 1 & otherwise \end{cases} \tag{2.7a}$$

$$g(z) = \frac{m_1}{\exp\left[(m_2 - z)m_3\right] + 1} \exp\left[m_4 (z - m_5)^2\right] \tag{2.7b}$$

where the m terms are parameters fitted to the total Si-O binding energy that was calculated for different coordination values using *ab initio* calculations. z is the coordination number of the O atoms, which uses a modified version of the Tersoff's cutoff function to smooth the cutoff value

$$z_i = \sum_{j=Si} \begin{cases} 1 & r < R - D \\ 1 - \frac{r-R+D}{2D} + \frac{\sin\left[\pi(r-R+D)/D\right]}{2\pi} & R - D \le r < R + D \\ 0 & r \ge R + D \end{cases} \tag{2.7c}$$

In addition to this bond softening function, the choice of parameters has been opened up from what is typically used with SW potentials. First, the cutoff distance values that are used in both the two- and three-body terms are no longer constrained to being the same value. Also, the $\cos\theta$ value is not restricted to being just the -1/3 for perfect DC. Despite these changes, the Si-Si-Si interaction is nearly identical to the original SW Si behavior, with only one parameter that has been modified. Other researchers have used the Watanabe potential with all of the original SW parameters showing no major differences and allowing the bulk Si behavior to be identical to SW for reference [81].

Despite lacking any ionic terms, this potential has shown remarkably good behavior for Si-O systems. The structural energies calculated for various Si-O clusters with changing bond lengths show an excellent fit to molecular orbital theory. In addition, five of the most common silica polymorphs are shown to be stable and to have lattice energies that are consistent with the expected stability. Finally, the initial paper [76] and subsequent work by the same authors and others [77, 81, 82] have shown that an amorphous oxide layer can be grown on the surface of pure Si that is not only stable, but consistent with what is experimentally observed.

2.4 Simulation Design

All work was done using the LAMMPS molecular dynamics simulation code [83] with a timestep of 1 fs. The temperature of the systems was constrained by using a Nose-Hoover thermostat [84].

2.4.1 Sphere Compression

Four sphere sizes were used with 5, 10, 20 and 34 nm diameters (3265, 26167, 209121, and 1027987 atoms respectively). The 5 nm sphere was analyzed at 0, 300 and 600 K, while the 10 nm and 20 nm spheres were studied at 0 and 300 K. All spheres were compressed along the [100] crystal direction. During separate simulations the 5 and 10 nm spheres were also compressed along [110] and [111] directions.

The particular version of the Tersoff potential used here is the third of the originally published parameters [21], which were chosen to give the best fit to the elastic properties of the diamond cubic phase. For these reasons, this potential is widely used for MD simulations of mechanically deformed silicon. Two versions of the Stillinger-Weber potential were investigated: one using the parameters from the original paper [18], and one where the system's energy had been scaled to better match real silicon [65].

Initial sphere creation was accomplished by generating all of the perfect bulk crystal lattice positions within a geometrically defined sphere. Following this, the sphere was annealed at 400 K for 1000 ps to relax the surface atoms before quenching down to 0 K.

Two planar indentation potentials were used to compress the spheres, one placed above the sphere (related to the direction of the y-axis) and the other equidistant below. This potential applies a force onto atoms according to their coordinates as given by

$$F_y(y) = ck(y - y_i)^2 \tag{2.8}$$

where F_y is the force in the y direction that the indenter applies onto each atom, k is a constant (taken to be 10.0 eV/ Å), y is the y-coordinate for that atom, and y_i is the y-

coordinate of the indenter. To insure that both indenters apply a repulsive force when they contact the sphere, for the upper indenter $c = -1$ when $y \geq y_i$ and $c = 0$ when $y < y_i$, whereas for the lower indenter $c = 1$ when $y \leq y_i$ and $c = 0$ when $y > y_i$. The applied load for the indenters acting on the sphere, P, is then found by summing this indenter force value over all atoms ($P = \Sigma F_y$).

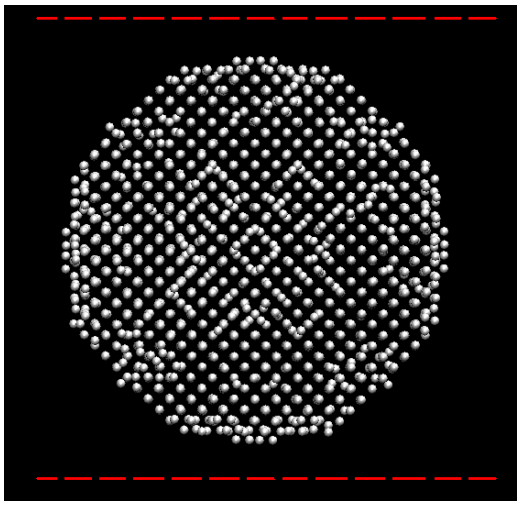

Figure 2.1: A side view image of a 5 nm silicon sphere. The dashed lines represent the indenter potential planes prior to loading.

Both indenters utilized inward velocities of 0.003125 Å/ps = 0.3125 m/s, resulting in a total displacement rate that is double this value. In order to save computation time, this rate was doubled for the 34 nm sphere. The indenters were allowed to compress the spheres until an engineering compressive strain (total displacement/diameter) between 0.4-0.6 before the spheres were unloaded at the same rate. During compression, the linear and rotational momenta of the total sphere were subtracted from each atom to prevent the sphere from rotating and drifting before and during compression.

2.4.2 Dislocation Interactions

Edge dislocations were introduced into the system by removing partial planes of atoms from the middle of the system. The removed atoms were at the center of the system with respect to the x direction and within the top and bottom ¼ of the y direction leaving behind two gaps in the crystal each a Burgers vector in width. The system was then compressed with a strain of -5% in the x direction and held for 1000 MD timesteps allowing the sides of the gaps to fuse together leaving behind perfect edge dislocations. The strain in the system was then relaxed. During the MD relaxation, the outer 10 Å of atoms in the x direction on both sides of the system were constrained and held at their perfect crystalline positions to discourage the dislocations from moving and leave a nice layer of atoms for the next step. Periodic boundary conditions were also imposed in the y direction.

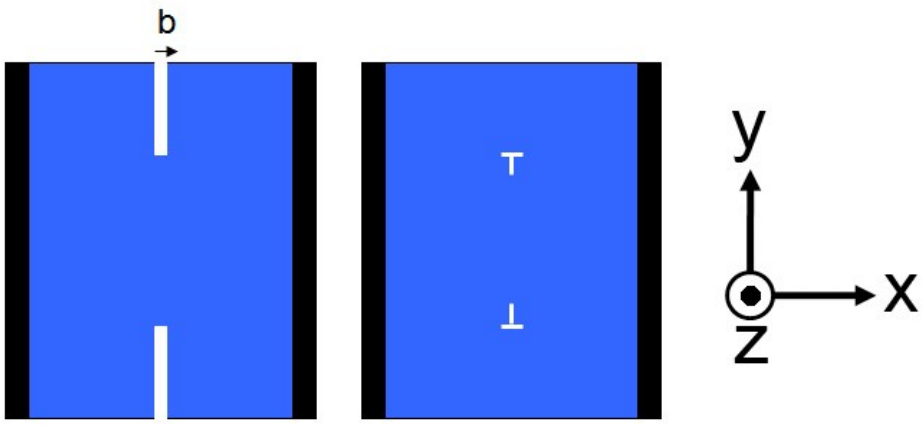

Figure 2.2: A schematic showing how the dislocations were created in the samples. First cuts were made in the material removing a Burgers vector of planes from the top and bottom 1/4 of the simulation. The simulation box was then relaxed by compressing it in the x-direction and allowing the free planes to join resulting in perfect edge dislocations.

Different simulations were set up to study how the dislocations interact with different surfaces. For the first type of simulation, periodic boundary conditions were applied in the x direction in order to imitate a bulk material. A second simulation was performed for which the two edges in the x direction were traction-free. All of the other simulations then involved introducing the oxide into the system by matching up the atoms that were constrained during the oxide growth with those that were constrained during the dislocation formation.

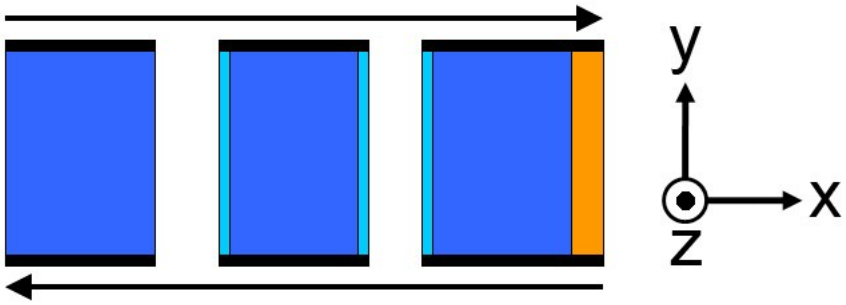

Figure 2.3: The different simulation systems explored. (a) Periodic conditions in the x-direction. (b) Free surfaces in the x-direction. (c) Oxide added to one or both surfaces in the x-direction.

To grow the oxide, first a number of oxygen atoms are randomly scattered onto the free surface forming bonds with the silicon atoms there. After relaxing the system with MD timesteps, the routine locates the silicon atom that is within bonding distance of 1-3 oxygen atoms and is positioned closest to the free surface. Once that silicon atom has been identified, its nearest neighbor silicon atoms are observed looking for a neighbor that is not saturated with oxygen's. Then an O atom is positioned between the two Si in such a way that they are both 1.6 Å from the O and the O is at least 1 Å from any other O. This position is chosen to minimize any local energy spike due to the

additional atom being too close to any other atom. The system, now containing one additional atom, is then relaxed for 1000 timesteps before the next O atom is added.

This routine can essentially be seen to provide a leap frog method of oxidation. Instead of waiting for diffusion to move the oxygen atoms near the surface into the silicon so more oxygen can react with the surface, the routine places the new oxygen atoms in positions where the existing oxygen atoms would be likely to diffuse. This routine still allows for the proper layer-by-layer growth while avoiding the time associated with diffusion. It also allows for the oxide to form as an amorphous material without starting with a particular oxide crystal structure.

To create the various simulations, a block of perfect silicon containing 9480 atoms was constructed. This block was oriented with its x-, y-, and z-axes oriented with the <110>, <1-11> and <-11-2> crystallographic directions and having dimensions of 85.846 Å by 94.068 Å by 26.606 Å respectively. As this size and direction allowed for all three directions to be perfectly periodic, this block then served as the basic building blocks for all of the simulations. The simulations that did not have an oxide were created in four sizes by adding a number of the basic block together along the x and y directions resulting in simulations that were 2X, 3X, 4X and 5X of the original block in both directions.

The oxide was also grown using this same base block. The oxide routine was run to oxidize one of the sides in the x direction with periodic conditions in the y and z directions. The atoms at the other x surface were held fixed in their perfect lattice positions. The oxidation routine was cycled until over 2000 oxygen atoms were added resulting in an amorphous layer of approximately 1 nm. Once created, this oxide

containing block could be easily adhered to the dislocation containing systems by matching up the atoms at the x surfaces that were held fixed during the dislocation formation with the fixed atoms during the oxidation. This allowed for the oxide to be simply attached to the systems minimizing the initial stress interactions. This allowed for simulations of various sizes to be examined without requiring the oxide to be grown on each separate system. However, it also means that there is some periodicity in the oxide due to the limited block and simulation sizes. It also has the further complication that for a corresponding simulation size, the dislocations are farther from the oxide interface than they would be from the free surface.

Alternative simulation designs were also created by merging the oxide block with the dislocation by placing non-fixed atomic planes next to each other. This allowed for systems where the initial distance between the dislocations and the oxide interfaces were comparable to the distance between the dislocations and the free surfaces of the oxide free systems. As a consequence, the atoms in the planes which were merged together were of higher energy for a short period of time until the planes relaxed.

The simulations performed to study the dislocation motion were accomplished by removing the previous constraints and either allowing the system to relax with no additional constraints, or holding the top and bottom 10 Å in the y direction fixed. When the top and bottom y surfaces were fixed, it constrained the simulation allowing for the shear strain of the system being controlled. In particular, simulations were done either holding the y boundaries at the initial positions, or moving them. For the simulations with moving boundaries, most were run at 300 K for 1000 timesteps to help

relax the system before the constrained atoms were displaced at a constant rate in the x direction. This resulted in a constant strain rate being applied on the system.

After simulating, the positions of the dislocations were identified by using the potential energy of the atoms to locate the defect atoms at the dislocation core. The position was then estimated by finding the center of mass in the x and y directions for these identified atoms.

Growth of the oxide was accomplished through a routine developed by Dalla Torre, et al. [82]. This routine mimics the layer by layer growth of oxide on a silicon surface. The routine was employed to form the oxide because it is practically impossible to simulate diffusion on the timescales associated with MD simulations without raising the temperature artificially high.

2.5 Phase Definition

One of the challenges with using molecular dynamics to identify phase transformations is determining a proper way of identifying and characterizing the different crystallographic structures. Basic crystallography identifies a crystal as consisting of a periodic lattice that can be represented by a basic unit cell that is repeated indefinitely. However, molecular dynamics involves a limited number of atoms, so unless periodic boundaries are used, no infinite crystal can be created thus limiting the overall size for which a crystal or grain can reach. This brings up the question of how small a region can be before it is no longer periodic enough to be considered a crystal structure.

Furthermore, most of the older commonly used potentials, such as the Tersoff and Stillinger-Weber potentials, are functions involving only nearest neighbors. Thus the energy associated with a given atomic arrangement only depends on the local bond structure and not on any larger scale periodicity. This makes it relatively easy and useful to the understanding of why the simulations are acting in a particular way by finding a method to characterize the bond structure. However, while a particular crystal structure may be composed of atoms all with identical bonding arrangements, simply having all the atoms within a system have that bonding arrangement does not guarantee that the resulting overall structure is the same, or even periodic.

Further complications to the understanding of the atomic structures seen in these simulations and experimentally arise from the use of the term amorphous material. Simply defined as any structure that lacks long term order, it broadly encompasses a wide variety of possible arrangements. The disorder present within these amorphous regions can vary greatly. On the highly disordered end, the structure can be so damaged that there is no uniform bond structure present. At the other end of the spectra, the local structure and bond order can be consistent throughout, but the long range order is disrupted or distorted.

The lower end amorphous description brings up some important questions in terms of molecular dynamics simulations of structures. As MD simulations offer snapshots in time, the local atomic deformation associated with an applied strain can be directly observed. This can allow for images of highly elastically deformed atomic structures being observed. If the strain is not uniform throughout, large gradients can form deforming some regions more than others, thus slightly disrupting the overall long

range order. Now, if the damage is purely elastic, then removing the applied strain should allow it to revert back to the perfect crystallographic structure. But if plastic damage has occurred, it could result in residual strain being present within the elastic region, leading to the inevitable question of whether this region classifies as amorphous or not.

2.6 Characterization Techniques

Phase identification was aided using a parameter related to the angles between the bonds of all of the nearest neighbors. This "angular" parameter was taken to be [85]

$$\frac{1}{N_b}\sum_{j=1}^{N}\sum_{k=j+1}^{N}(\cos\theta_{ijk}-\cos\theta_{DC})^2 \qquad (2.9)$$

where N is the number of nearest neighbors that atom i has, θ_{ijk} is the angle between atoms i, j, and k, θ_{DC} is the bond angle for bulk diamond cubic, and N_b is the number of bond angles that have been summed. In essence, the difference in the cosine of the bond angle with respect to the perfect diamond cubic structure is squared, and then averaged for all bond pairs around a given atom. For the results presented here, a cutoff of 3 Å was used to determine the nearest neighbors included in this expression. This angular parameter is dependent on each atom's coordination number, but is advantageous as it can distinguish between multiple phases and defects that have the same coordination number. The parameter's formulation is related to the three-body term of the Stillinger-Weber potential [18].

Dislocation activity was monitored primarily with the slip vector parameter [86], given by

$$\vec{S}_i = -\frac{1}{N_s}\sum_{j\neq i}^{N}\left(\vec{R}_{i,j} - \vec{R}^0_{i,j}\right) \qquad (2.10)$$

where N is the total number of nearest neighbors to atom i, N_s is the number of neighbors that are on an adjacent slip plane to atom i (e.g., $N_s = 1$ if slip occurs on an {111} diamond cubic lattice), $\vec{R}^0_{i,j}$ is the vector from atom i to its neighbor j at an initial unstrained reference configuration, and $\vec{R}_{i,j}$ is the corresponding vector at the current configuration. By finding the relative displacement of the nearest neighbor atoms j with respect to a given atom i, it can be determined if a plane neighboring atom i has slipped and in what direction. Dividing by $-N_s$ scales the vector's magnitude so that it will be equal to the Burgers vector of the dislocation that caused the slip. Appendix One contains the code of the program used to analyze the sphere compressions, which includes the calculation of the angular and slip vector parameters.

Figure 2.4 shows both the angular parameter and the slip vector in action. The colored spheres show the positions of the atoms with angular values between 0.12 and 0.16. As the angular parameter is not independent of the coordination number, nearly all of the atoms with angular values within this range also had a coordination of 5. The thin white arrows show the slip vectors that have magnitudes of approximately 3.84 Å corresponding to the Burgers vector of a perfect dislocation. A cluster of similarly colored atoms with angular values around 0.13 is visible in the lower left corner of the image revealing a region that had transformed into BCT5. Near the upper right of the image is a line of atoms colored by angular values between 0.13-0.15 corresponding to the atoms at the core of a dislocation. This shows that even though the BCT5 region and the atoms at the dislocation core both have coordination number 5, the angular

parameter is able to distinguish between the two. The white arrows going from the bottom of the image up to the dislocation core then show the path that the dislocation had traveled, along with the direction of the Burgers vector for that dislocation.

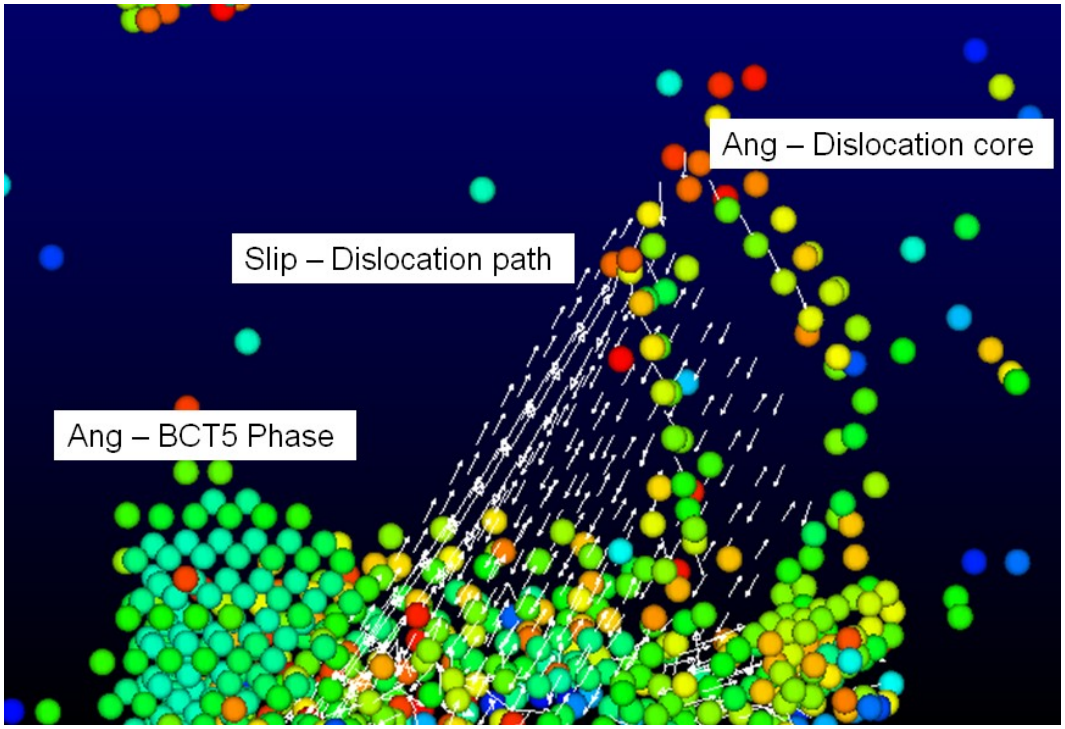

Figure 2.4: An image showing how the angular and slip vector identification parameters work. The colored spheres represent atoms with angular values between 0.13-0.15 allowing for the core of a dislocation and a region of BCT5 to be identified. The white arrows show the direction of the slip vectors corresponding to atoms that have slipped approximately one Burgers vector from their neighbors indicating the path that the dislocation had traveled.

Chapter Three: Tersoff Potential and Phase Transformations

The focus of this chapter is on the results obtained from the simulations that used the Tersoff atomistic potential. While both this chapter and the next focus on compressions of silicon nanospheres, it is important to separate the results obtained from the two different potentials as they result in different atomistic behaviors. As previously mentioned, the Tersoff potential was designed to best represent the pressure induced phase transformations in silicon. Therefore it is an ideal choice for investigating how the formation of β-Sn affects the measured mechanical properties.

3.1 Contact Area Analysis

One of the great unknowns associated with the experimental study of nanoparticles under compression is the contact area between the indenter and the particle. This contact area is necessary to know in order to accurately calculate the hardness of the nanoparticles and the associated stresses within during any applied load. However, there is no method for directly measuring this contact area experimentally. The specific shape and dimensions of the indenter tip can be worked out using the techniques developed for nanoindentation, but the same cannot be done for the particles themselves. The only "direct" experimental method is to use images from *in situ* TEM to visually measure the contact area, assuming that the tip is directly centered over the particle. With this sole exception, the contact area must be assumed to fit one of the models that have been developed.

So far, four different models have been investigated for use with experimental results for predicting the contact area. These four models are referred to as the Hertzian model [87], the geometric model [7], the cylindrical model [47], and the harmonic mean [88] and are given by the equations

$$a_H^2 = \frac{1}{2} R\delta \qquad \text{Hertz;} \qquad (3.1)$$

$$a_G^2 = R\delta - \frac{\delta^2}{4} \qquad \text{Geometric;} \qquad (3.2)$$

$$a_C^2 = \frac{4R^3}{3(2R-\delta)} \qquad \text{Cylindrical;} \qquad (3.3)$$

$$a_{HM} = \frac{2}{a_H^{-1} + a_C^{-1}} \qquad \text{Harmonic mean.} \qquad (3.4)$$

Within these models, a is the radius of the contact area, δ is the measured displacement, and R is the sphere radius. The Hertzian model is the elasticity solution for a sphere contacting a flat plane at low strains. The Geometric model is the area of intersection between a sphere and a plane. The Cylindrical model is the contact area associated with the compression of a cylinder. The use of the cylindrical model is based upon the concept that after sufficient deformation, the spheres can be approximated as cylinders. Finally, the harmonic mean is calculated using the Hertzian and Cylindrical models to average between the low strain Hertzian behavior and the high strain cylindrical behavior.

Molecular dynamics simulations offer a unique opportunity for investigating the contact area models and determining a more refined fit. The specific positions of all the atoms within the system are known along with the position of the indenters allowing for

the contact area to be numerically calculated. Through the atomic interactions, the simulations account for both the elastic and plastic responses of the nanoparticles at low and high strains. It also accounts for any drastic changes in shape that the current geometric based models cannot handle. Of course, any fits to the data are still approximations based upon the assumptions that the particles are initially perfectly spherical, and that the material response given by the potential is adequately realistic.

The radius of the contact area, a, between the spheres and the plates was calculated by identifying the atoms (N_a) that were within 0.1 Å from each of the indenters. By identifying the center of mass ($X_{C.M.}$ and $Z_{C.M.}$) for that collection of atoms, the contact radius was found with a formula given by Vergeles et al. [89]

$$a^2 = \frac{2}{N_a}\sum_{i=1}^{N_a}\left[\left(x_i - X_{C.M.}\right)^2 + \left(z_i - Z_{C.M.}\right)^2\right] \tag{3.5}$$

This measured contact radius is shown in Figure 3.1(a) for the 20 nm sphere and Figure 3.1(b) for the 10 nm spheres at 0.01 K. Along with measured contact radius squared are the predicted values for both the elastic Hertzian model and the Geometric model. Initially, at low loads the contact radius shows a discrete nature as its value stays constant for a number of steps before jumping to a larger value. See Figure 3.1(a). This appears to be nothing but a geometrical artifact of the sphere design and indenter potential used. The top and bottom-most atomic planes of the spheres consist of only a handful of atoms resulting in the initial contact area being relatively small. On continued loading, the outermost atomic layers are pushed far enough into the sphere that the indenters then come into contact with the next inward planes of atoms. When

this occurs, the contact area suddenly increases. The effects of this jump in contact area decreases with increasing sphere size and displacement.

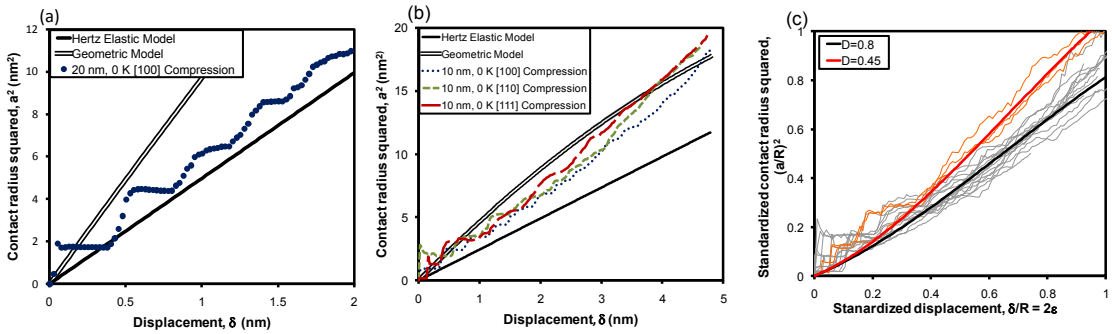

Figure 3.1: The squared contact radius vs. displacement. (a) At low displacements for the 20 nm diameter spheres (0.01 K shown), the contact area jumps due to the surface steps resulting in discontinuous behavoir, but still loosely follows the Hertz model. (b) As seen with the 10 nm diameter data at 0.01 K, at moderate displacements, the curves fall directly between the elastic and the geometric models. At high displacements, the contact area begins to increase at a higher rate deviating from both models. The high displacement deviation is more profound for the 300 K compressions (not shown here). (c) The contact radius squared for all of the data shown with the new model fit from Equation (3.4).

Figure 3.1(a) shows that for the 20 nm diameter sphere at small displacements, the contact radius behavior still contains the discontinuous jumps, but is smooth enough to show a general trend following the Hertzian model. This is the expected behavior as only elastic behavior is seen within the sphere during this period. At moderate displacements (~1-3.5 nm for the 10 nm spheres in Figure 3.1(b)), the measured values for all three orientations lie between the Hertzian and geometric values, with the change in contact radius with displacement closer to the Geometric model. While neither model gives a "best fit" to the measured contact radius, they can be seen to offer an upper and lower bound during this displacement range. The largest displacements (Figure 3.1(b)) reveal that the spheres no longer follow either model as the contact area

begins to increase at a greater rate than either model predicts due to neither model accounting for expansion of the sphere in the uncompressed directions.

From these comparisons, it is possible to develop a more accurate, but yet still simple model for the contact area behavior. One method would be to linearly mix the two theoretical models. This results in an expression such as

$$a^2 = \frac{(D-\delta)a_H^2}{D} + \frac{\delta a_G^2}{D} = \frac{1}{2}R\delta + \frac{R\delta^2}{2D} - \frac{\delta^3}{4D} \qquad (3.6)$$

where D is a fitting constant with units of distance. This equation is a third-order polynomial fit containing only one unknown that given the appropriate choice of D fits the previously mentioned behavior of the measured contact area for all displacements observed here.

Conceptually, at first glance Equation (3.6) appears to increase the relative amount of plastic to elastic behavior as the displacement increases. However, as neither of the base models accounts for the sphere's incompressibility at large displacements, this conceptual argument is faulty. This new model gives an empirical fit to the data even though it is derived from the two mathematical models.

As seen in Figure 3.1(c), there is considerable variation in the values measured for the contact area under the different conditions studied here. However, comparing Equation (3.4) to the measured values reveals that a D value of 0.8 R offers a decent fit to nearly all of the data. The only clear exceptions are the [111] compressions at elevated temperatures, which favor a D value closer to 0.45 R. As the constant D is fitted to the entire strain range, this variation due to the loading direction could depend on either changes in the plastic behavior or anisotropy in the elastic constants and

crystal structure. With either value of D, the general trend given by Equation (3.4) better predicts the measured contact area than either the Hertzian or Geometric models for the full displacement range between $0 \leq \delta \leq R$.

Higher strains were also investigated with one simulation of the 10 nm sphere compressed along the [100] direction at 300 K. Figure 3.2 shows the contact area measured for this sphere along with all of the models discussed previously. It is observed that the Hertzian and Geometric models, which do not account for the incompressibility of the spheres, reach a maximum contact radius equal to the radius of the sphere. Because of this, the linear mixing also suffers from this limitation and is inadequate for compressions over 0.5 strain. The cylindrical model is seen to always overestimate the contact area at all strains. As for the harmonic mean, it is seen to be comparable to the geometric model at low strains but remains consistent with the measured simulation data up to roughly 0.5 strain. Above this it does deviate from the measured data, but it still has the correct trend of sharply increasing. From this, it is seen that the harmonic mean does offer the best fit to the measured contact area.

With the high strain data, another attempt was made to better model the contact area. Even though the cylindrical model overestimates the contact area at all strains, it still follows the general trend in behavior. Therefore, an appropriate model for the spherical compression could be obtained by modifying the expression for the cylindrical compression. One possible modification was found where the cylindrical model is simply multiplied by the strain to the 1/3 power. This allows for the contact radius to be zero at zero strain and decreases the modeled contact radius at higher strains. It is seen that this simple adjustment provides a good fit up to a strain of 0.7.

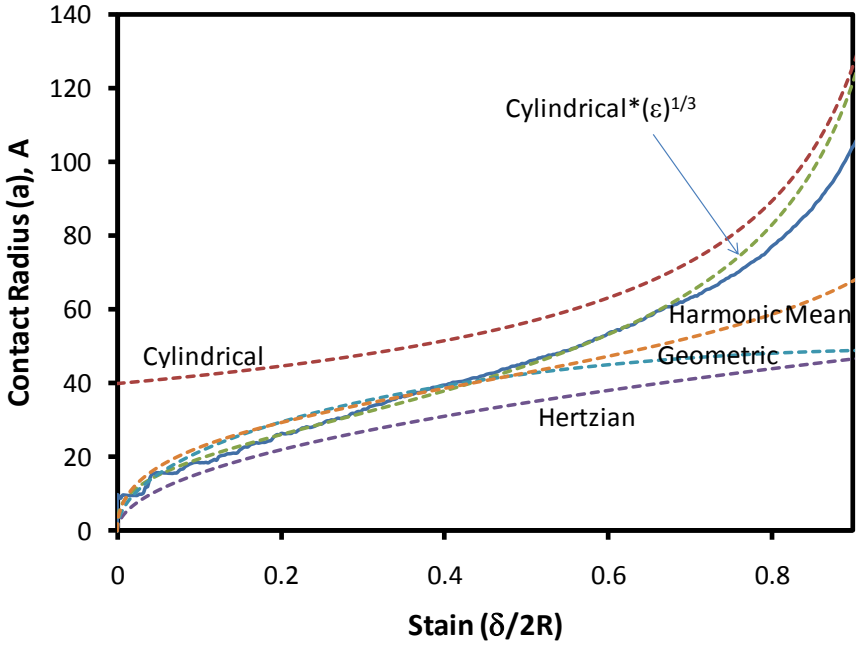

Figure 3.2: The measured contact area out to a strain of 0.9 compared to various contact radius models.

3.2 Measured Load and Stress

The load vs. displacement curves obtained from the 10 nm diameter spheres are shown in Figure 3.3. Similar compression behavior is observed in the 5 and 20 nm diameter spheres, which are not shown. For a given orientation, the load at small displacements is nearly identical at the different measuring temperatures corresponding to the spheres behaving elastically. Following this, the higher temperature runs deviate from the 0.01 K curve (e.g., for the 10 nm spheres the 300 K load is less than the 0 K load for displacements greater than 1-2 nm.) This indicates that yielding has occurred in the higher temperature runs resulting in a substantial change in the loading rate. For the 0.01 K curves, the [100] compression shows a similar drop in slope indicating yielding followed by a sharp peak in the applied load at high displacements, whereas

the [110] and [111] compressions continue to increase fairly steadily throughout the loading with the occasional drop indicating some form of yielding. These two behaviors result in the maximum load near 0 K being approximately 2 times greater than the maximum loads at the higher temperatures for all three orientations.

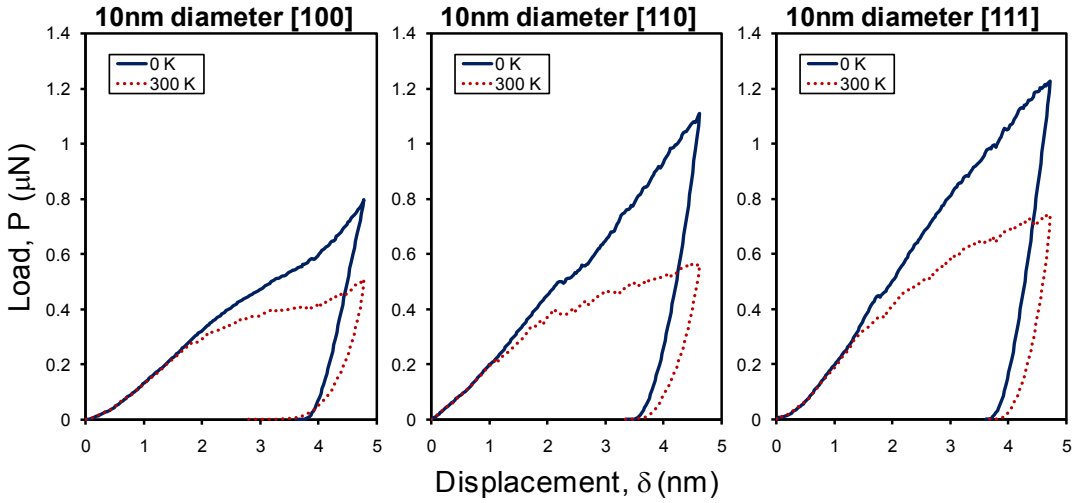

Figure 3.3: The load vs. displacement curves for all 10 nm simulations. Note that the maximum force is nearly twice as high for the 0.01 K simulations than it is for the other higher temperature simulations.

The averaged contact stress ($P/\pi a^2$) was also calculated for all of the samples and is shown in Figure 3.4 for the 10 nm spheres plotted vs. compressive strain of the sphere ($\delta/2R$). The general behavior of the contact stress on increasing strain was that one large peak or a series of smaller peaks would initially appear at low strains corresponding to the jumps in the radius of the contact area mentioned above. After these peaks, the stress values would drop before beginning to linearly increase again indicating that the material is still behaving elastically. Between strains of 0.1 and 0.2, the behaviors of the 0.01 K and higher temperature simulations are seen to deviate from each other. The contact stress in the 0.01 K runs plateaus and remains close to the

maximum value reached, while the 300 and 600 K runs have stresses that reach a maximum value before steadily decreasing with additional strain indicating a softening behavior.

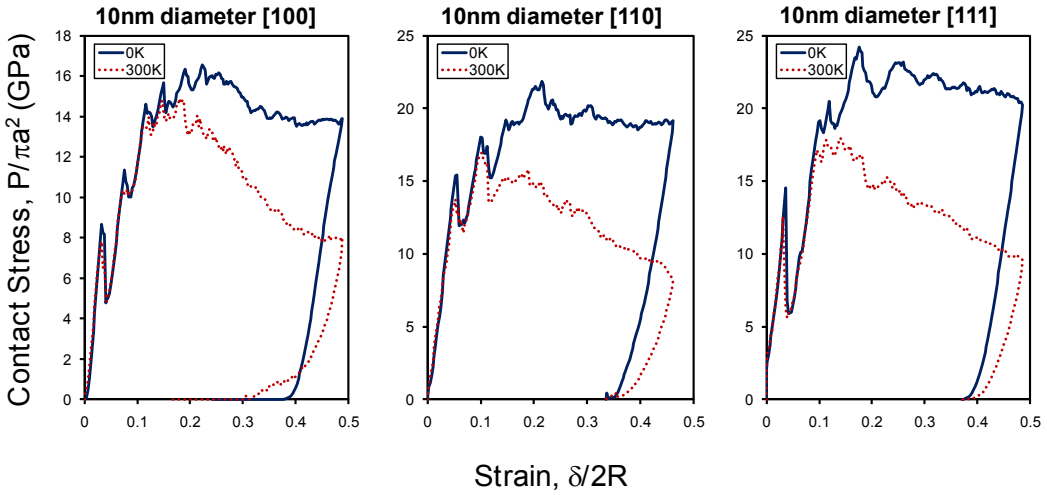

Figure 3.4: The contact stress vs. engineering strain curves of the 10 nm diameter simulations.

The hardness values reported from the experimental results [7] were taken to be the maximum load for a given compression divided by the calculated contact area. This definition is identical to that of the averaged contact stress here allowing the two values to be compared directly. Experimentally, for a single load-unload, the hardness for bulk silicon is around 12 GPa while the reported hardness values for spheres varying between 19-46 nm in radius and unknown orientations were 10-40 GPa [7]. Repeated loading showed an increase in hardness suggesting that the spheres become harder as the load increases, or as the plastic damage increases.

The maximum stress values measured from these simulations do not vary by much between the temperatures: 16-24 GPa for the 0.01 K compressions vs. 14-22 GPa for the 300 and 600 K compressions. However, the contact stress values at the

maximum strain are quite different, with 14-20 GPa at 0 K compared to 7-9 GPa at 300 K. This matches with the difference in the load vs. displacement curves in Figure 3.2, where the maximum load at 0.01 K is approximately twice what it is at 300 K.

The maximum contact stress values from the simulations are a decent match to the experimental hardness values, with values ranging from the bulk silicon value to roughly 2 times the bulk value. A discrepancy is seen, however, in that from the experiments, the hardness increases slightly as the load increases, whereas the 300 K simulations show a marked decrease in the contact stress upon increased displacement. This indicates that the simulations presented here using the Tersoff potential at ambient temperatures fail to give the large hardness values associated with the experimental spheres of this size.

The stiffness was also calculated for the 10 nm diameter spheres at the maximum displacement by evaluating the initial slope of the unloading curves from the stress vs. strain plots. At 0.01 K, the values for unloading modulus for the [100], [110] and [111] directions respectively were 154, 236, and 252 GPa, while at 300 K were 103, 117 and 144 GPa. These show that the simulations performed at 0.01 K were 50 to 100% stiffer at their maximum loadings than the similar runs at 300 K.

3.3 Observed Phase Transformations

The lattice and elastic constants for DC, BCT5 and β-Sn were calculated near 0 K and are included in Tables 3.1, 3.2 and 3.3 respectively. In regards to BCT5, the energy and structure is seen to be slightly closer to the first principle calculations with the Tersoff potential than with the Stillinger-Weber potential. Using the Tersoff

potential also produces a very good agreement for the structure of β-Sn. In addition, it has been previously shown that the Tersoff potential correctly predicts the pressure and resulting volume changes associated with the transition from DC to β-Sn [65].

Table 3.1: The lattice parameter and elastic constants for the diamond cubic structure of silicon.

	Tersoff [65]	Tersoff (This work)	Experimental [65]
a (Å)	5.432	5.432	5.429
E (eV/atom)	-4.6297	-4.63	-4.63
C11 (GPa)	142.5	139.7	167
C12 (GPa)	75.4	74.1	65
C44 (GPa)	69	69.1	81

Table 3.2: Lattice constants and elastic constants for the BCT5 phase.

	Plane wave pseudopotential [17]	Stillinger-Weber [17]	Tersoff (This work)
a (Å)	3.32	3.3544	3.298
c (Å)	5.97	6.5148	6.468
E (eV/atom)	-4.41	-4.24	-4.419
C11 (GPa)	144	415	162
C12 (GPa)	124	243	108
C13 (GPa)	45	139	76
C33 (GPa)	160	208	205
C44 (GPa)	35	40	49
C66 (GPa)	63	101	143

Table 3.3: The lattice constants and elastic constants for the β-Sn phase of silicon.

	Tersoff [65]	Tersoff (This work)	Experimental [11]
a (Å)	4.905	4.903	4.686
c (Å)	2.57	2.568	2.585
E (eV/atom)	-4.3027	-4.3023	
C11 (GPa)		297	
C12 (GPa)		60	
C13 (GPa)		39	
C33 (GPa)		378	
C44 (GPa)		36	
C66 (GPa)		29	

A study of the ideal positioning of the atoms in different silicon phases allowed for a determination of appropriate values of the angular parameter for each phase. From this, diamond cubic, BCT5 and β-Sn have angular values of 0, 0.12 and 0.18 respectively. It should be noted that this value is but a measure of how far the bonds around a given atom are from the perfect diamond cubic structure and elastic strain will affect the value. However, as phases will have particular bond orientations their ideal structures will have specific angular values. An atom that has an angular value similar to the ideal value for a given structure and the correct coordination number can then be said to have the bonding characteristic of that phase. While the angular value might not be unique to a given phase, i.e. diamond cubic and hexagonal diamond both read as 0, it is a useful asset in identifying regions that can be examined closer for phase confirmation or in obtaining an estimate of the amount of a given phase that is present.

When compressed along the [100] crystalline orientation at 0.01 K, the 5 and 10 nm spheres first behaved elastically, then yielded by disordering at the high stress regions near the contact areas. Cross sections of the 10 nm sphere are seen in Figure 3.5(a, b). Further compression resulted in the disordered region growing outward and surrounding the core. Small clusters of BCT5 were identified within the disordered regions. When the displacement was high enough that the disordered regions created by the top and bottom plates reached each other, the core region began transforming to β-Sn. The β-Sn core continued to grow up to unloading and remained after unloading. The final structure was seen to consist of three layers: a β-Sn core surrounded by a disordered region, which in turn was surrounded by elastically deformed DC.

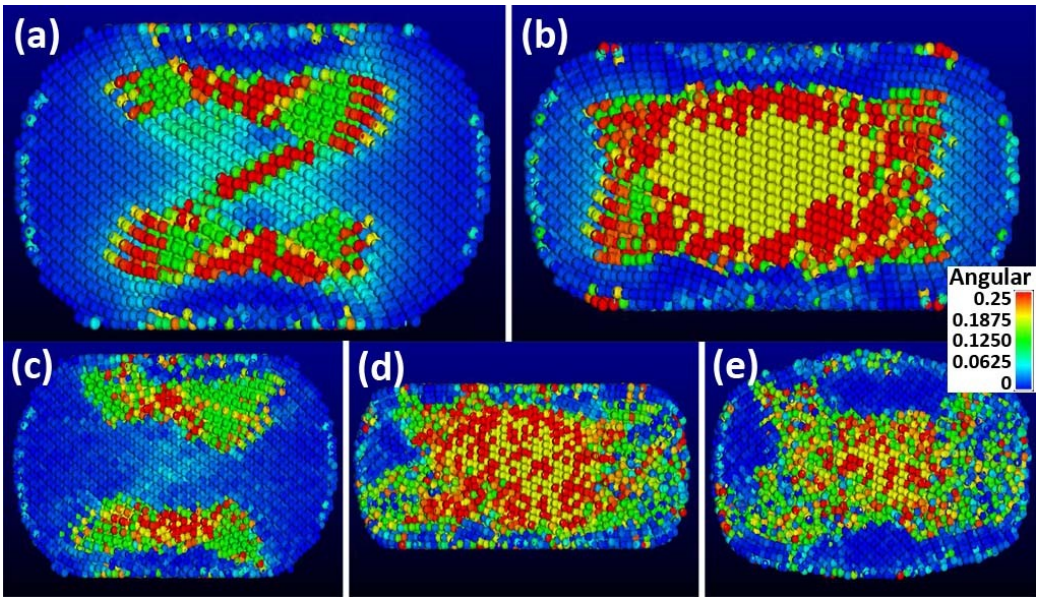

Figure 3.5: Cross sectional images using the angular parameter to highlight the phase changes seen during [100] compression of 10 nm diameter spheres. (a) 0.01 K compression at 3.3 nm displacement resulting in green (mid grey) regions of BCT5. (b) 0.01 K after unloading showing a yellow (light grey) β-Sn core. (c) 300 K compression at 2.9 nm displacement. (d) 300 K compression at the maximum displacement (e) 300 K after unloading revealing that a large amount of the β-Sn had relaxed back to a four coordinated phase colored in blue (dark grey).

Figure 3.5(c-e) shows that for [100] compression at 300K in the 10 nm sphere, a similar yielding behavior resulted with disordered regions leading to the formation of β-Sn in the core of the sphere at large displacements. However, there was a noticeable increase in the scatter of the atomic behavior throughout both the diamond cubic and β-Sn regions due to thermal fluctuations. Increasing the sphere size to 20 nm in diameter still resulted in regions of β-Sn, but the morphology differed by showing it initially forming close to one of the contact areas as opposed to the sphere's center.

Upon unloading, all of the [100] compressed spheres at 300 K exhibited extensive relaxation as the β-Sn began to revert to a 4 coordination structure, visible in Figure 3.5(e). To better characterize the relaxed region, a radial distribution function,

RDF, was calculated for the atoms for the last recorded timestep of the 10 nm sphere (the same timestep shown in Figure 3.5(e)). RDF was used as it has been shown to allow for a distinction between DC and the other relaxed silicon phases commonly referred to as Si-III (bc8) and Si-XII (r8) [24]. All three of these phases have a coordination number of 4 (using a cutoff distance of 3 Å) but can be distinguished from each other by having different next nearest neighbor distances which would show up as distinct peaks within the RDF. Figure 3.6 shows the RDF for only the atoms with a coordination number of 4 revealing 2 broad peaks that correspond only to DC (2.35, and 3.84 Å) and no distinct peaks characteristic of either Si-III or Si-XII (3.2-3.45 Å). The broadness of the peaks suggests that at this timestep, the region is disordered/deformed classifying it as an amorphous phase that is close to the DC structure.

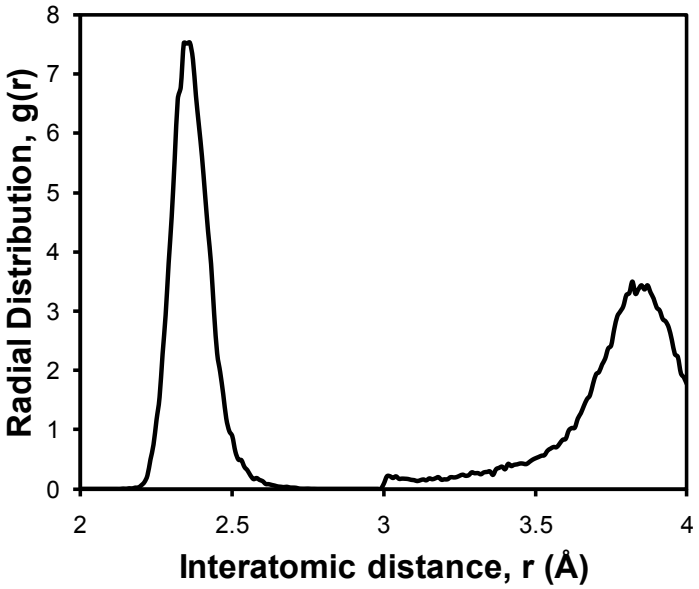

Figure 3.6: Radial distribution function for the 4 coordinate relaxed phase of the final timestep of the 10 nm [100] compressed sphere at 300 K. Only peaks for DC at 2.35 and 3.84 Å are seen and no distinguishing peaks are seen between 3.2 and 3.45 Å that would indicate one of the other known relaxed silicon phases. The nearest neighbor cutoff value used to isolate the DC phase was 3 Å resulting in the sudden step at this value.

Interestingly, almost no β-Sn was seen resulting from compression in either the [110] or the [111] directions. Cross sections of these compressions in the 10 nm spheres are given in Figure 3.7. Besides the lack of β-Sn, the most notable aspect of these results is the vast difference in behaviors at the two temperatures. For the [110] loading, the 0.01 K runs (Figure 3.7(a-b)) showed large amounts of elastic strain prior to the amorphous yielding compared to the 300 K runs (Figure 3.7(c-d)). In fact, the elastic strain for the [110] orientation at 0.01 K was high enough that the effective coordination number increased to 6 at the high stress regions prior to yield (Figure 3.7(a)). However, the greatest difference in behavior due to changing temperatures is seen in the [111] compression tests. At 0.01 K, the high loads applied while the sphere is still elastic allows for the resulting yield to occur throughout the entire sphere as shown in Figure 3.7(e). At 300 K, yield occurs almost immediately upon contact and remains localized near the indenter-sphere interface leaving the center nearly undeformed shown in Figure 3.7(f).

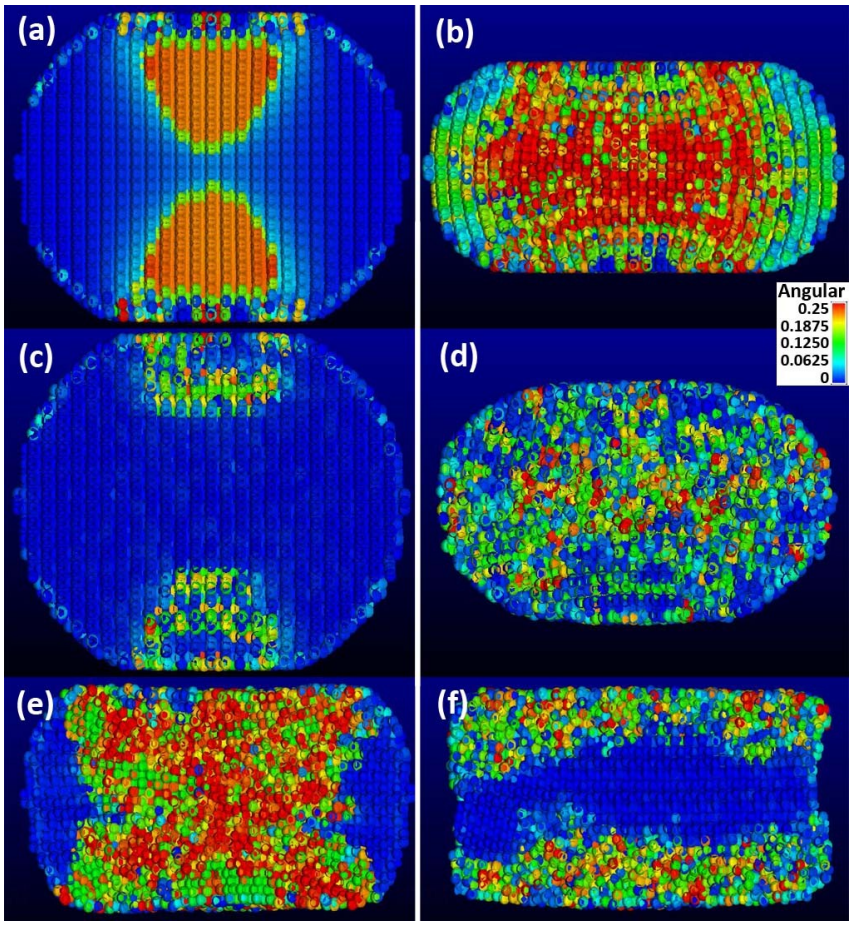

Figure 3.7: Cross section images of 10 nm diameter spheres colored with angular values. (a) 2.5 nm displacement of [110] compressed sphere at 0.01 K resulting in regions of high elastic strain that appear different with the angular parameter due to the measured coordination increasing to 6. (b) Post compression of the [110] 0.01 K sphere with no distinguishable regions of a particular phase. (c) 1.7 nm displacement of [110] compressed sphere at 300 K showing that yield occurs much earlier than at 0.01 K preventing the extensive elastic behavior seen in (a). (d) Post compression of the [110] 300 K sphere also with no distinguishable regions of a particular phase. (e) Post compression of the [111] 0.01 K sphere showing disordered material throughout. (f) Post compression of the [111] 300 K sphere revealing disorder only at the surface near the contact areas.

The coordination number and the angular parameter values were also used to quantify the amounts of the different phases present. This was accomplished by counting the number of atoms with the correct coordination number for a particular phase along with an angular value within a range about the ideal value. For instance, β-

Sn was taken to be atoms that had 6 nearest neighbors and an angular value between 0.17 and 0.25. Although this analysis produces a specific value, it should be considered as a rough estimate as it does not take long range order or elastic strain into account.

Results in Figure 3.8 show that for [100] compression, the β-Sn quickly increases in concentration at the higher strains and approaches nearly 10% at the maximum strain. During the initial unloading, the elastic strain is released from the sphere resulting in the measured β-Sn value increasing for a short period as strained β-Sn relaxes to its ideal configuration. Following this, the β-Sn in the 0.01 K simulation levels out whereas the 300 K simulation shows a marked decrease due to the reverse phase transformation. The values calculated for the [110] and [111] orientations show *only 1 and 2 % respectively for the atoms reaching the criteria for β-Sn*, confirming the results of the visual analysis.

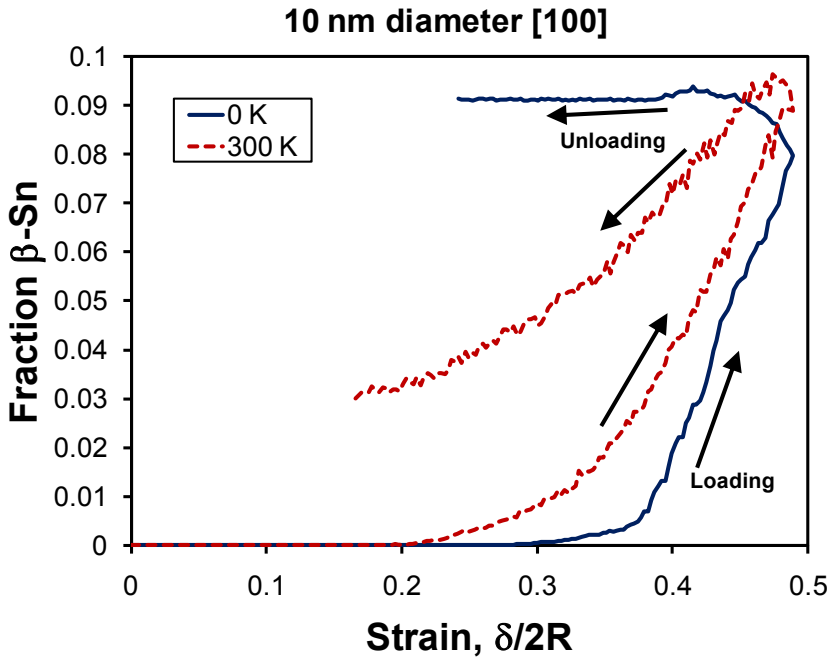

Figure 3.8: Fraction of β-Sn plotted vs. strain for the 10 nm diameter [100] compressed spheres.

The concentrations of DC and BCT5, along with β-Sn as a function of strain are shown in Figure 3.9 for the [100] compressed 10 nm sphere at 300 K. Prior to loading, only 90% of the atoms in the sphere register as DC. The 10% classified as other at this point is due to the surface atoms being excluded for not having 4 nearest neighbors. At a strain of 0.2, the fraction of DC is seen to begin to rapidly decrease, while a small amount of BCT5 begins to form. The first β-Sn forms soon after this, but appreciable amounts do not appear until roughly a strain of 0.3. The BCT5 concentration decreases slightly when β-Sn forms. After a strain of 0.4, β-Sn continues to increase steadily while both BCT5 and DC decrease. At the maximum strain, 45% of the material is not classified as being one of the three phases with β-Sn occupying roughly 9%, BCT5 roughly 13%, and DC roughly 33%. Upon unloading, an initial increase in all of the values is seen as the sphere elastically relaxes allowing more atoms to be counted as a particular phase. Further unloading shows the amount of DC increase and the amount of β-Sn decrease due to the reverse phase transformation.

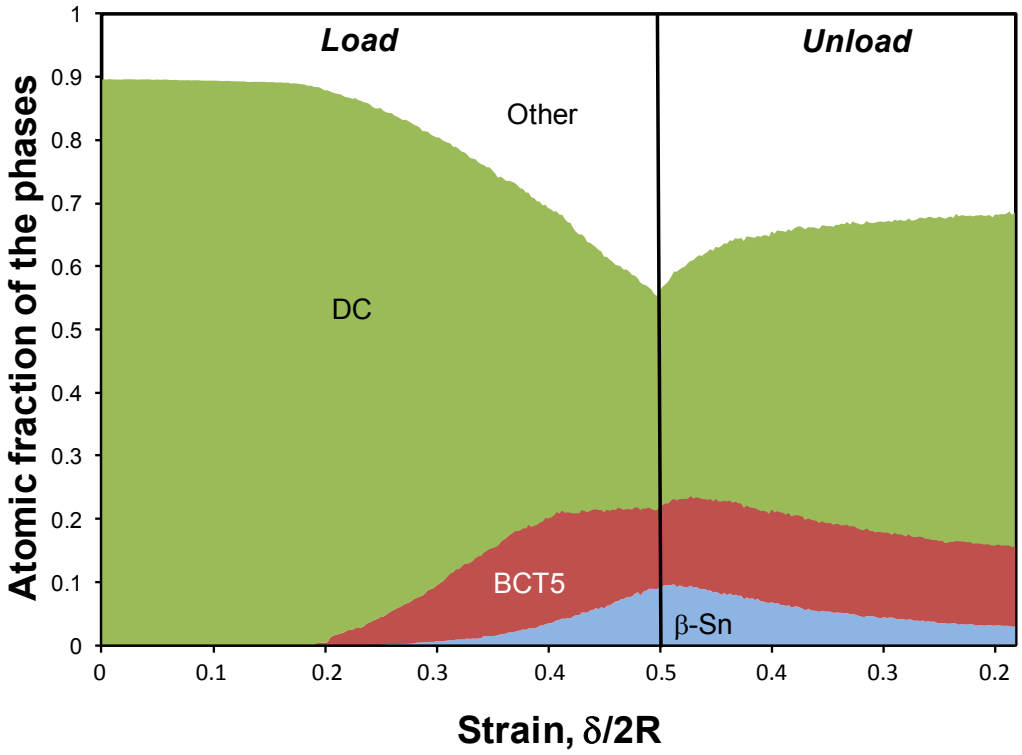

Figure 3.9: The progression of the estimated fractions of the different phases as seen within the 10 nm diameter [100] compressed sphere at 300 K. The solid vertical line marks the maximum displacement, after which the sphere is unloaded.

Together, the results suggest a different explanation for the hardening behavior seen by Valentini et al. [60] for Si sphere compression modeled with the Tersoff potential. For all orientations, the 0.01 K compressions show that very large contact stresses are reached and the applied load is steadily increasing all the way up to unloading. In contrast, the 300 K compressions initially behave elastically, but then clearly yield with little hardening afterwards. Furthermore, the β-Sn transformation was seen during the [100] compression at both 0.01 K and 300 K, but was not present during compression for the other orientations. This suggests that the high contact stresses are independent of the β-Sn phase transformation. Instead, it appears that these high

contact stress values are due to the Tersoff potential having a high resistance to plastic yielding for 0.01 K simulations. Since yield stress generally scales with modulus, this would be consistent with the observations in Figure 3.4 and of the unloading slopes at 0 K being greater than those at 300 K for all three orientations even when little β-Sn is present.

Even though the β-Sn transformation is widely accepted as occurring experimentally during indentation of flat silicon surfaces in different orientations [27-29, 31], no direct evidence of the transformation, either as elbows in the unloading curve or the presence of other phases in a diffraction pattern, has been observed for small compressed silicon nanoparticles [4, 7, 44, 49]. This was for spheres less than about 100 nm in diameter. The simulations presented here show that β-Sn will only form within the nanospheres when compressed along the [100] crystallographic direction. From the experimental results, the particles' orientations are unknown with respect to the compression direction and assumed to be random due to the fabrication technique [7, 44]. Therefore, it follows that many of the nanoparticles will not undergo the β-Sn transformation as their orientation is not favorable. Conversely, if the orientation is the only decisive factor, it is possible that some of the particles would show evidence of the transformation.

As one representation of the β-Sn structure is as a tetrahedral compression of the DC structure along one of the <100> directions, it makes sense that [100] compressions result in the formation of this phase. However, for the other orientations, the behavior is different within the compressed spheres than it is for indented bulk silicon suggesting that the spherical geometry is less favorable to the formation of β-Sn. The most notable

difference between the two geometries is that the sphere is much less constrained in the directions normal to the applied load than an indented flat surface is. This allows for the sphere to easily expand in these directions, as seen with the contact area analysis. Because of this the hydrostatic pressure within the spheres will be less for a given loading than bulk indentation potentially making the β-Sn transformation less likely. It also follows that the formation of β-Sn during compressions of the other orientations may become more likely as the particle size increases. As this is a qualitative assessment, future work would be necessary to determine exactly how these stress states differ.

It should also be noted that there are considerable differences between the simulation and experimental conditions. The largest simulations presented here are half the size of the smallest particles that have been experimentally compressed. In addition, the rate of compression is quite different: Nowak et al. reported an experimental displacement rate of 10 nm s^{-1} resulting in compression runs around 10 seconds [49], whereas the displacement rate of 6.25×10^{8} nm·s^{-1} for the simulations resulted in total compression runs lasting around 10-20 nanoseconds. Either of these factors could greatly influence the mechanical response and make different yielding mechanisms more favorable.

3.4 Hardening at High Strains

The works by Valentini, et al. [60] and Zhang, et al. [61] find large stresses and hardness values associated with higher strains than what were investigated in sections 3.2 and 3.3. In particular, both papers find that there is a pronounced hardening during

strains of 0.4 to 0.6 in which large regions of β-Sn are present within the core of the compressed spheres. Therefore a hardening behavior may be associated with β-Sn at high compressive strains.

To explore this possibility, data obtained at higher strains was examined. In addition to the 5 nm spheres which were compressed up to strains of 0.6, the 10 nm sphere was also compressed along the [100] direction at 300 K up to complete compression. The average compressive stresses along the compression direction within the spheres were calculated by averaging the per atom stress obtained from the virial formula over the entire sphere size. This allowed for a direct comparison with the results given by Zhang, et al. [61]

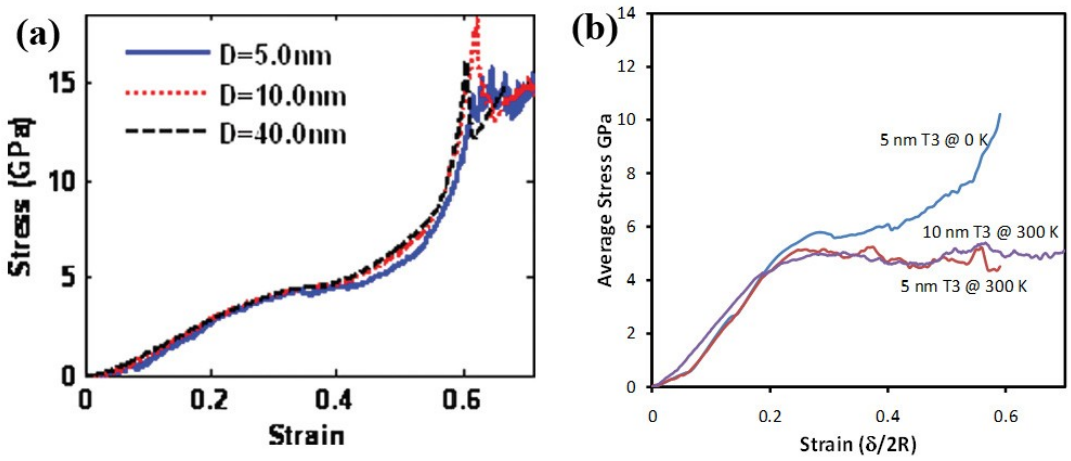

Figure 3.10: (a) The average stress vs. strain previously reported for a variety of Si Tersoff sphere simulation sizes. (b) A comparative plot of the stress vs. strain obtained during this work showing a clear difference in behavior near 0 K and at 300 K. Note that the 0.01 K plot is consistent with the plots in (a).

The average stress values show that the 5 nm sphere compressed at 0.01 K is consistent with what was found by Zhang, et al. for all of the sphere sizes that they

investigated. In particular, a sharp increase in the average stress is observed within the 0.4 to 0.6 strain range. However, the simulations at 300 K revealed a completely different behavior with the contact stress remaining fairly constant around 5 GPa after the initial yielding.

The contact stress was also calculated for all strains within the 10 nm sphere (Figure 3.11). The maximum hardness associated with the compression of this sphere at 300 K was roughly 15 GPa and was reached shortly after yield. This differs drastically from the 25-34 GPa values previously reported as occurring around a strain of 0.65.

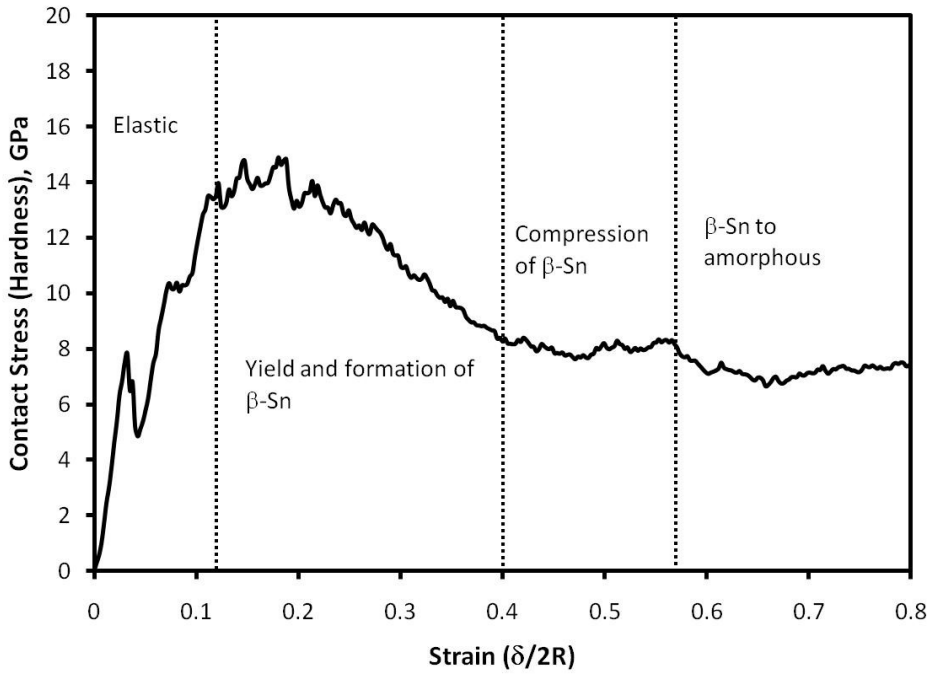

Figure 3.11: Contact stress vs. strain for the [100] compressed 10 nm sphere at 300 K. The maximum hardness is observed just after yield. Regions of distinct hardening/softening behavior are seen to correspond to the material's response.

The contact stress vs. strain plot (Figure 3.11) did reveal that the morphology of the sphere influenced the measured hardness values. Initially, the contact stress increases as the sphere behaves elastically. After yield, however, softening occurs as the DC structure breaks down and gradually transforms into β-Sn. No hardening is seen to occur with the formation of β-Sn. At a strain of about 0.4, the contact stress plateaus indicating a region of consistent or slightly increasing hardness. During this period, the core of the sphere has already transformed into β-Sn resulting in a change in hardening because the material being compressed is now β-Sn as opposed to DC. Finally, at strains around 0.6, the β-Sn begins to yield and turn amorphous as well.

What these results indicate is that hardening behavior can be associated with the presence of a large volume of β-Sn within the silicon nanospheres. However, this hardening does *not* result in a large increase in the measured hardness during this range. A simple linear extrapolation of the behavior from the 0.15-0.40 strain region showed that if that softening trend continued, the hardness at 0.6 strain would be around 4 GPa as opposed to the measured 8 GPa. While this is noteworthy, the difference is small compared to the 20 GPa measured for the 5 nm sphere at 0.01 K for the same strain. Once again, the large hardness values are associated with simulating at low T as opposed to the presence of β-Sn.

Even though hardening is observed with the Tersoff potential, it fails to explain the experimentally observed hardening and hardness values. Experimentally, hardening and high hardness values are measured for strains as small as in the range of 0.1-0.2. This cannot be explained as due to the presence of large regions of β-Sn as even the

compression direction most favorable to β-Sn formation does not show the associated hardening until 0.4 strain. Therefore, the experimental hardening must be due to some other mechanism.

3.5 Dislocations

In addition to the phase transformations, dislocation yielding was also observed in three of the simulations: 5 nm 600 K [100] and [110] compressions and 20 nm 300 K [100] compression.

Distinguishing dislocations within the 5 nm 600 K simulations was difficult due to the small number of atoms and the high thermal scatter. However, hints of dislocation motion were observed by plotting the direction of the slip vectors of the atoms. By restricting the plot to only showing atoms with slip vector magnitudes around what is expected for perfect dislocations, small planar regions were seen to contain parallel slip vectors. As the slip vector indicates the region where a dislocation with a particular Burgers vector equal to the slip vector has passed through, finding a planar region where all of the atoms have nearly identical slip vectors indicates that a dislocation, however short lived, had existed there. Both of the [100] and [110] compressions contained one of these dislocation hints, with the [100] shown in Figure 3.12. Since no dislocations were observed in the 5 and 10 nm spheres at 0.01 K and 300 K, a clear temperature dependence on dislocation nucleation is evident at this size scale.

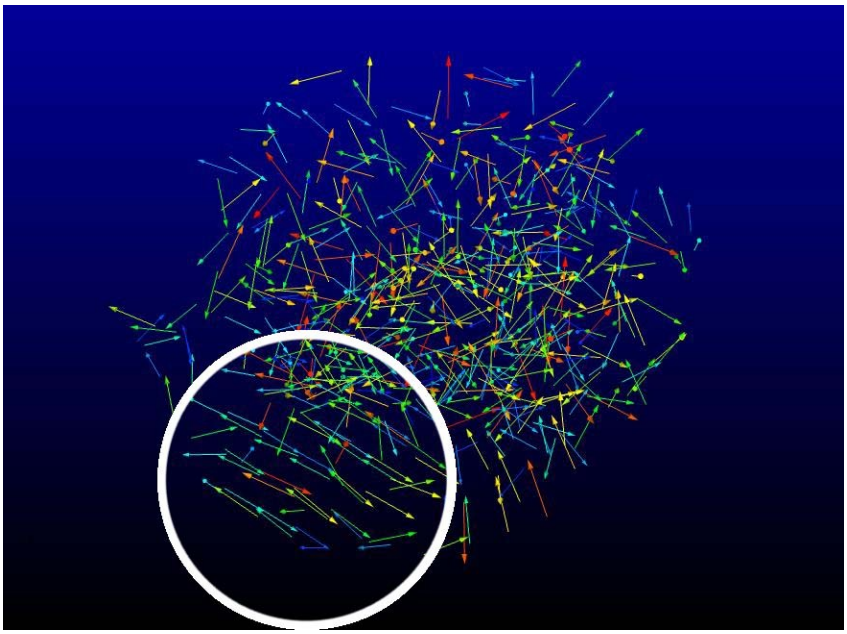

Figure 3.12: Image of the slip vector direction for atoms with slip magnitudes close to a perfect dislocation within the [100] compressed 5 nm diameter sphere at 600 K. The circle indicates a planar region where the slip vectors are oriented nearly parallel to each other indicating the presence of a dislocation.

For the 20 nm diameter 300 K simulation, a total of 9 dislocations were observed prior to unloading. The first of these dislocations is shown in Figure 3.13. The presence of these dislocations indicates that there is a cutoff size for dislocation formation between 10 and 20 nm diameters at 300 K. Close analysis of these dislocations revealed them to be perfect 1/2 <110> shuffle set dislocations. They were seen to form at high strains when there was already substantial regions that had deformed to an intermediate state between the DC and β-Sn phases. Slipped planes indicating the motion of these dislocations appeared at the edge of the disordered regions going to the surface of the sphere.

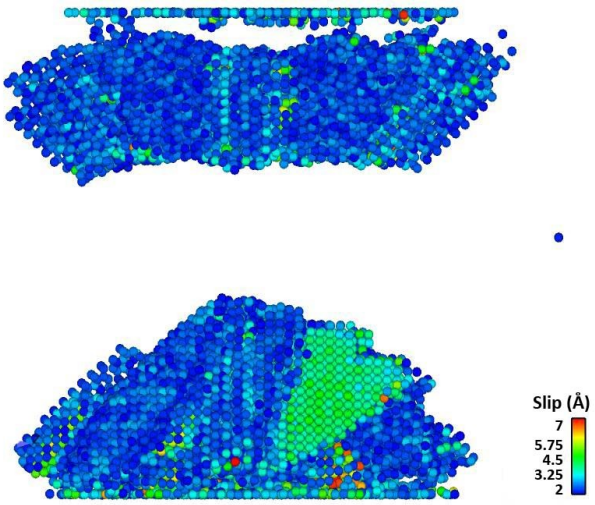

Figure 3.13: Atoms colored with the slip vector magnitude for [100] compression of a 20nm diameter sphere at 300 K. The blue (dark) regions of low slip (2-3 Å) has a volume type shape representing phase change-type deformation resulting in regions of BCT5 and β-Sn. The green (light) region shows a plane that has slipped due to a full 1/2<110>{1-11} dislocation.

The fact that dislocations can be observed in these MD simulations of silicon nanospheres allows for a comparison with the hardening theory proposed in the experimental papers. In short, this theory states that the high hardness values are the result of the buildup of dislocations confined within the small particle dimensions. Due to the presence of an oxide layer on the surface of the sphere, any dislocations that form within the sphere are delayed from terminating on a free surface leading to dislocation pileups. These pileups potentially result in a back stress opposing the applied load resulting in high hardness values. As the particle size decreases, the volume that the dislocations are confined within also decreases allowing for higher stresses to be reached [49]. It is also believed that this behavior will have a lower limit when the particle passes a critical size that is necessary for dislocations to form [7].

Only the 20 nm diameter sphere compressed at 300 K showed multiple dislocations, confirming that there is a critical size for dislocation nucleation. However, no accompanying hardening behavior occurred as the dislocations all formed individually and quickly disappeared upon reaching the surface as no oxide barrier was present. There was never more than one dislocation present within the sphere at any timestep thus no dislocation interactions or pileups.

Witnessing dislocation hardening and interactions using molecular dynamics would require that multiple dislocations be present within the sphere at the same time. To obtain this, dislocations must form more readily and/or must be impeded from reaching and disappearing at the sphere's surface. The former could be possible by changing the atomic potential used to one that is more prone to dislocation behavior, while the latter can be accomplished by introducing a surface barrier representative of the oxide layer. Also, increasing the sphere size would accomplish both of these conditions as only the largest sphere showed dislocation behavior and further increases in size would mean that any dislocations that formed would have to travel further to reach the surface. Incorporating an oxide layer and increasing the sphere size would make the simulations more comparable to the experimental results, but doing either would require an increase in the complexity and computational time. Changing the potential could result in an overall simulated behavior that is less realistic than what is presented here, but if it is capable of showing multiple concurrent dislocations in sizes comparable to those here, it would offer a simple and efficient way of studying the dislocation interactions. Chapters 4 and 5 in this thesis investigate these possibilities.

3.6 Conclusions

The yielding properties observed within Tersoff modeled silicon nanospheres showed a high dependence on changes in temperature, orientation and sphere size. For three temperatures and three orientations, β-Sn is only seen to appear in substantial amounts during [100] compression. Only the simulations near 0 K show high contact stress values indicating that the chosen atomic potential fails to accurately model the experimental large hardness values at ambient temperatures. The observed large contact stress values near 0 K are attributed to a high yielding point, resulting in elastic behavior at large strains and a stiffening behavior. As the β-Sn transformation observed in these simulations forms only for specific compression orientations and any hardening associated with the β-Sn phase is observed at high strains, it is highly unlikely for the β-Sn to play a role in the experimentally observed hardening.

Dislocation behavior is also identified and indicates that a critical size and temperature must be exceeded for dislocations to nucleate. Although no hardening results from the dislocations in these simulations, it is proposed that the larger dimensions and the presence of an oxide on the experimentally observed particles would result in a greater number of dislocations to be present within the particle potentially allowing for dislocation hardening.

Chapter Four: Dislocations with the Stillinger-Weber Potential

This chapter continues the investigation of the yielding and hardening mechanisms within silicon nanospheres. Here, the atomistic potential used is the Stillinger-Weber potential resulting in completely different yielding mechanisms being predominant. In particular, a considerable number of dislocations are observed allowing for the analysis of how their interactions affect the measured hardness.

4.1 Plasticity Mechanisms

Compression along the [100] direction showed the formation and propagation of numerous full 1/2<110> dislocations through the spheres. The dislocations were seen to travel on a pair of shuffle set {111} planes connected by a {110} plane forming a "V" shaped slipped region (Figure 4.1). This geometric shape was seen to result from the dislocations nucleating on the {110} type planes oriented 45° from the applied loading, then cross slipping to the more crystallographically favored {111} planes.

Each dislocation would homogeneously nucleate in the region just below the contact area. These regions near the contact points experience stresses comparable to the ideal strength of the material. After nucleating, the dislocations then proceed diagonally through the sphere to the surface. Upon reaching the surface of the sphere, the dislocations were able to terminate resulting in visible surface steps.

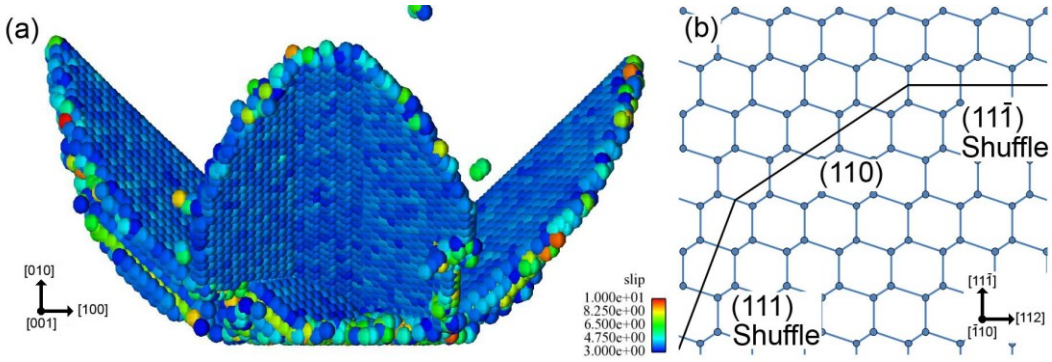

Figure 4.1: Slip vector images from the 20nm diameter radius spheres compressed at 300 K. The visible planes are ones that have slipped due to a dislocation that had traveled on that plane. The dislocation loops were seen to have traveled on at least two {111} planes connected by {110} planes.

While dislocations were the primary yielding mechanism for the [100] compression orientation they were not the only yielding mechanism observed as regions of BCT5 were formed. The simulation of the 10 nm sphere compressed at 0.01 K featured the most prevalent BCT5, which is shown in Figure 4.2. Atoms identified as having bonding consistent with BCT5 were shown to form in two distinct shapes: a conical structure and a highly directional region. The conical BCT5 region is seen to form within the high stress regions near one of the contact areas and forms as a volume as might be expected for a high pressure phase transformation. The conical region of BCT5 remained throughout the compression and subsequent unloading. In contrast, the directional region is composed of subsequent {110} planes that have been sheared. This builds up a region consistent with BCT5 plane by plane. At higher stresses, full dislocations nucleated from the edge of the directional BCT5 region and grew to the surface. Further increases in the displacement resulted in that particular BCT5 region disappearing. An in depth discussion of the formation of the BCT5 regions and how

they relate to the dislocations nucleating on the {110} planes is addressed in Section 4.2.

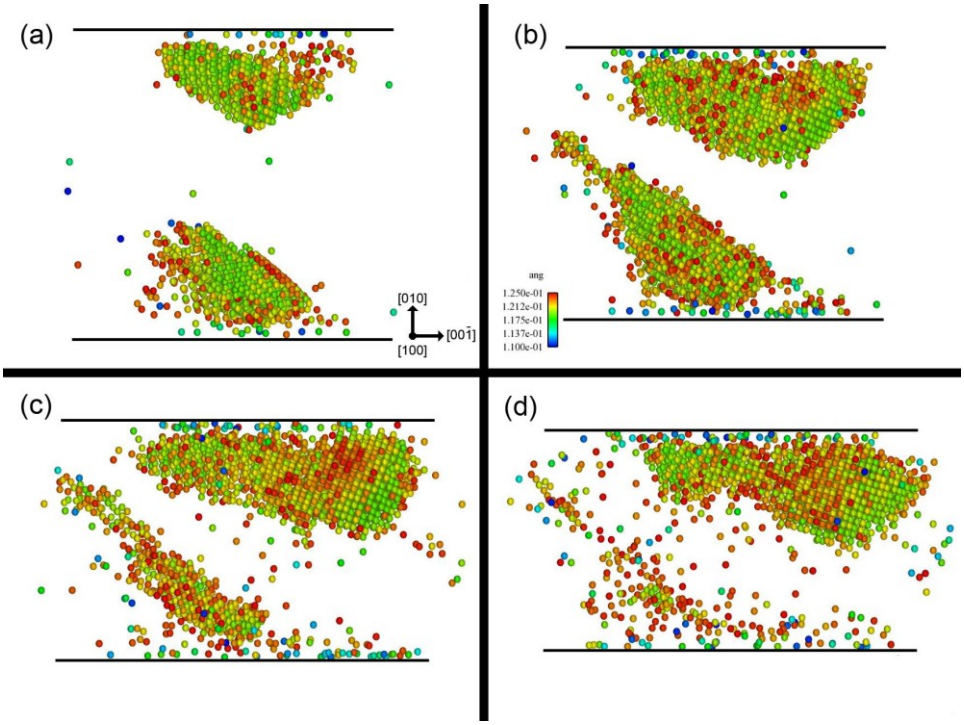

Figure 4.2: Images showing the BCT5 yielding behavior seen at 0 K for compression along the [100] direction within a 10 nm diameter sphere. Lines have been added showing the positions of the indenters. The applied displacement increases from (a) to (d) with respective displacements of 2.06, 3.09, 3.86 and 4.39 nm. In (a), two distinct morphologies of BCT5 are seen: conical (top) and directional (bottom). As the load increases, the conical region is seen to expand and move off center while the directional region first grows and then disappears as it is replaced by full dislocations.

Compression along the [110] direction shows the formation of a wedge shaped yield zone just within the sphere near both of the compression zones that appears to be a complex region of partial dislocations, BCT5 and amorphous silicon. Continued loading shows that partial dislocations nucleate and connect the wedge shaped regions at the top and bottom that quickly transforms into a stacking fault (Figure 4.3). With

further deformation, the stacking fault remains and grows slightly while the wedge-shaped regions quickly decompose into purely amorphous zones.

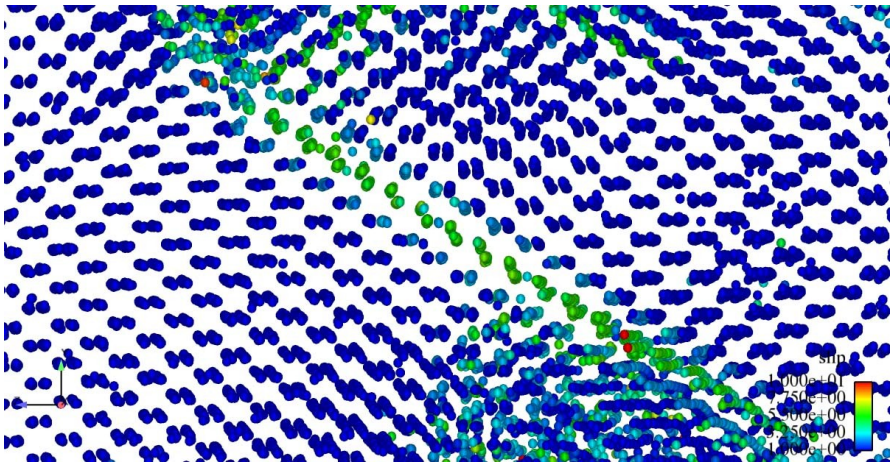

Figure 4.3: A cross-section of a [110] compressed Stillinger-Weber sphere clearly showing a stacking fault.

For the third direction tested, the [111] direction showed both dislocation and stacking fault behavior. Initially, dislocations would form at the contact points and grow into the material. But rather than travelling all the way to the sphere's edge, the dislocation line would transform into a stacking fault that connected to the surface that the dislocation had originated from (Figure 4.4(a)). At higher displacements, full dislocations would originate from this slipped region and proceed all the way to the opposite side and outside of the sphere (Figure 4.4(b)).

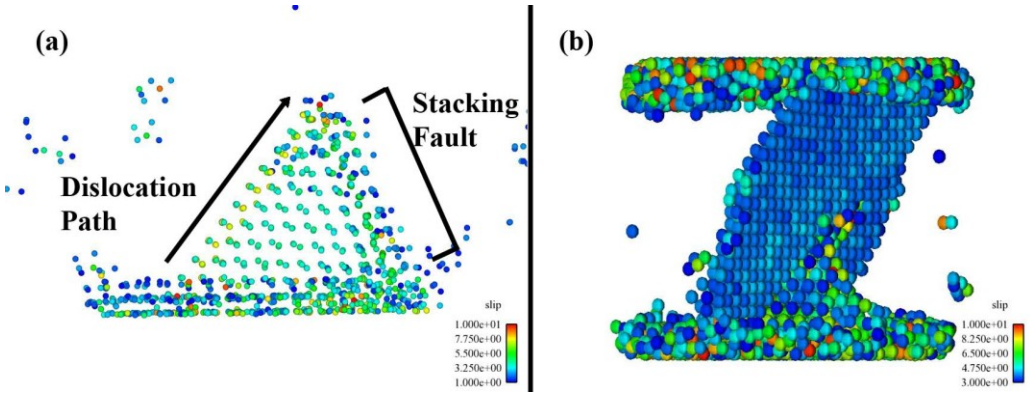

Figure 4.4: The dislocation behavior for [111] compression. (a) At low loads, a dislocation forms at the contact region and grows into the crystal and transforms into a stacking fault connecting the slipped region back to the amorphous contact area. (b) Higher loading results in the appearance of full dislocations that travel to the opposite surface.

When compared to the previous results using the Tersoff potential in Chapter 3, some major differences are observed. For 300 K simulations, dislocation behavior was observed even in spheres as small as 5 nm with the Stillinger-Weber potential. In contrast, the Tersoff results indicated a critical size for dislocation nucleation between 10 and 20 nm in diameter below which no dislocations form. In addition, no dislocations were seen to nucleate at 0.01 K for the Tersoff potential, but they still readily formed for the Stillinger-Weber potential. This shows that the Stillinger-Weber-potential has the distinct advantage over the Tersoff potential for MD studies of dislocations in silicon nanoparticles because smaller, less computationally expensive spheres readily show dislocation yielding. Therefore, any reasonable MD investigation of Si nanostructures that require large numbers of dislocations, such as studying dislocation interactions, would be better accomplished with the Stillinger-Weber potential.

4.2 Dislocation Nucleation on {110} Planes

Nucleation of the observed full dislocations during the [100] sphere compressions were seen to originate on {110} type planes. An examination of the {110} planes just prior to dislocation nucleation revealed the formation of an intermediate slip band. This slip band is visible as a plane of atoms in many of the characterization methods by having a slip vector value between 1 and 3 Å, an angular value around 0.13 and a coordination number of 5. Interestingly enough, these same values correspond to the regions identified as BCT5. This suggested that the two observed behaviors are not independent of each other and thus required a more in depth look.

In order to examine the intermediate slip that occurs prior to the nucleation of the full dislocations, a set of $(0\,\bar{1}\,1)$ planes from the 20 nm sphere compressed at 0.01 K were identified as having a region that slipped with respect to each other and this region was visually isolated from the bulk material. Images (Figure 4.5) were taken normal to the atomic planes prior to the intermediate slip, after the slip, and finally after further slip that resulted in the formation of the full dislocation at the edge of this region. Comparing the first and last images to each other, it can be seen that the dislocation is perfect and the slip occurred in the [011] direction. Analyzing the atomic positions revealed that the resulting dislocation had the expected burgers vector of a/2[011]. As for the intermediate step, it appears that the two planes are approximately halfway between the two end positions giving a slip of roughly a/4[011]. Note that this slip does not correspond to a crystallographically ideal stacking fault site.

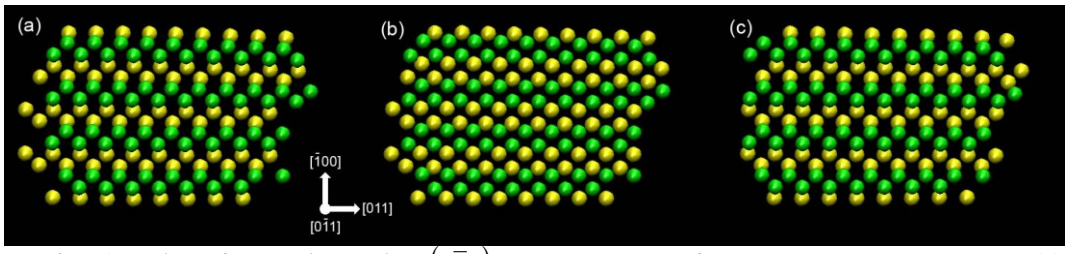

Figure 4.5: A region of two neighboring $(0\bar{1}1)$ planes extracted from the compressed sphere. (a) This image shows the planes prior to any slip occurring with all atoms consistent with the DC structure. (b) Increased loading resulted in this intermediate slip of approximately a/4[011] between these two planes. This state remains stable for a number of imaged steps. (c) Slip between the planes continues at higher loads resulting in the region returning to the DC structure. As the two planes in this region have been displaced exactly a/2[011] from their initial positions in (a), a perfect dislocation is formed at the edge of the region.

The fact that this intermediate state is stable for a number of timesteps suggests that it corresponds to a local energy minimum. To better quantify this, a generalized stacking fault (GSF) curve was calculated for slip between pairs of {110} planes and compared to the GSF curve for shuffle set {111} slip. This was accomplished with MD simulations in which incremental slip was applied between a pair of slip planes along the slip direction. The systems were designed according to the work done by Godet, et al. [70] such that the normal to the slip plane is oriented along the x-axis, periodic conditions are applied in the y- and z-directions and the system size is chosen to be just large enough to avoid issues with the periodicity and free surfaces. This resulted in systems of 2048 atoms and 61.448 Å × 30.724 Å × 21.724 Å dimensions for the {110} test and 1440 atoms and 47.0338 Å × 26.6064 Å × 23.0418 Å for the {111} test. The energy was then calculated for all of the incremental slip positions by constraining the atoms in the two slip planes only along the slip direction and performing a local minimization.

The resulting GSF curves (Figure 4.6(a)) calculated with the Stillinger-Weber potential show a clear minimum at a displacement of a/4 $<1\bar{1}0>$ for {110} shearing. Even more substantial is the fact that the slope and maximum seen with the {110} slip are only slightly higher than what is observed with the {111} shuffle set slip. The maximum values for the {111} and {110} slip measured are 0.051 and 0.058 eV/Å^2 respectively. The fact that the peak value of the {110} slip is comparable to the shuffle set slip indicates that dislocation nucleation on {110} planes to be an alternative low energy yielding path. This is indeed what is seen to occur during compressions along the [100] direction as the resolved shear stress is greatest along {110} planes.

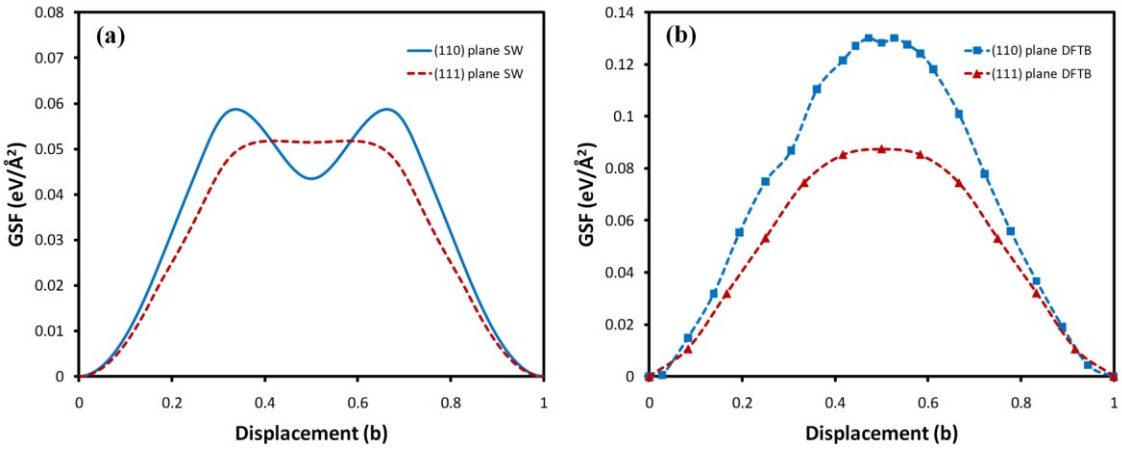

Figure 4.6: The generalized stacking fault curves for <110> slip in Si. (a) Using the Stillinger-Weber potential, there is a clear energy minimum at the halfway point for slip on (110) planes making the GSF energies comparable to slip on the (111) shuffle set planes. (b) Quantum mechanical calculations using DFTB reveal that there is a slight minimum, but the large difference between the maximum energies makes (111) shuffle set slip much more favorable.

This clearly shows that there exists a minimum. However, it does not explain why the intermediate state should be energetically favorable. The answer is obtained by a close examination of the atoms that shows that at least between the two planes, the

atoms have a coordination number of 5 and the bond orientations are very similar in appearance to those that appear in the BCT5 structure. As the potential energy calculated for a given atom with the Stillinger-Weber potential is only dependent on its nearest neighbors and BCT5 is a stable phase with this potential [17]. When the crystal is sheared and the coordination of the atoms increases to 5, the atomic arrangement rearranges to BCT5 bonding as the BCT5 structure is the favored 5 coordinated structure. In terms of crystallography, it would be incorrect to call it a region of BCT5 as it only occurs on two neighboring atomic planes, but at least locally the bonding around each atom in those two planes is the same as in the BCT5 regions.

From this, it can be proposed that the intermediate slip seen is a low energy yielding mechanism for the Stillinger-Weber potential at low temperatures and can lead to the formation of either BCT5 regions or the nucleation of dislocations. Once part of a plane has undergone this intermediate slip, further loading can result in more yielding to occur. If neighboring planes yield by this intermediate slip, then the region quickly develops into a volume of BCT5. On the other hand, if the slip continues on the same plane as the original slip, then the atoms will jump to the next perfect crystalline sites forming a full dislocation on {110} planes. While the exact stress conditions in the region of yield play the largest part in determining which behavior will occur, it is reasonable that increasing the temperature will make the dislocations more favorable than the BCT5 as there will be more energy to overcome the energy dip associated with the intermediate slip and reach the true crystalline position.

It is interesting to note that slip on {110} planes was also reported for [100] Stillinger-Weber Si nanowires in compression [57]. This indicates that the {110}

dislocations are not limited to nanospheres and may occur in any structures where the resolved shear on the {110} planes is larger than the resolved shear acting on {111} planes. Interestingly, only shuffle-set dislocations were observed with the Stillinger-Weber nanowires in tension. The {110} pathway may favor having a compressive pressure in the high stressed regions.

To better investigate this novel dislocation nucleation pathway, similar GSF curves were also calculated using quantum mechanical molecular dynamics. These calculations were performed by Dong-Bo Zhang. These simulations used the DFTB [90] method, as implemented in the computational package TROCADERO [91]. Different from most empirical many body potentials such as Stillinger-Weber type, DFTB treats the electrons in an explicit way allowing for a more robust and realistic representation. The DFTB calculations of the GSF curves were performed similarly to the Stillinger-Weber results with the exception that fewer atoms were used: 16 for the {111} plane and 32 for the {110} plane. This allowed for the smallest repeat units in the y and z directions while allowing for 8 and 16 planes respectively parallel to the imposed slip.

The GSF curves obtained with DFTB are shown in Figure 4.6(b). The behavior of the {111} shuffle-set slip plane is consistent with what was previously reported for this slip plane [92] with the exception being that the maximum measured here of 0.087 $eV/Å^2$ is less than the previously reported maximum of 0.11 $eV/Å^2$. This can be accounted for in that here the energies were calculated by relaxing the atoms at the slip planes in the two dimensions normal to the slip direction, whereas the original calculations only relaxed normal to the slip plane. As for the {110} slip plane, a slight

decrease in energy is observed at the halfway point along the slip direction. However, the maximum energy associated with slip in the {110} plane is 0.13 eV/Å^2, which is considerably larger than the energy associated with the {111} shuffle-set. As the quantum based simulations are more accurate, this suggests that the {110} nucleation is less likely than the {111} one. However, the size of the barrier indicates that the {110} nucleation is not prohibited to occur experimentally. Note also, that under large applied external strain conditions, the energetic barriers might be significantly lower than the value reported here.

4.3 Dislocation Yielding

All simulations presented here are for compressions of the sphere along the [100] crystallographic direction. The [100] compression orientation was studied as it was the orientation that was found to most readily show dislocations. It also offers a unique opportunity where the Burgers vectors for all of the 1/2<110> dislocations are 45° from the compression direction. With this information, a simple geometric estimate can be used to relate the total residual plastic displacement, δ_p, to the number of dislocations, *n*, with Burgers vector *b*

$$\delta_p = nb\cos(\theta) \tag{4.1}$$

where θ is the angle between b and the compression direction. This equation assumes that all of the permanent deformation is due to dislocations and that each dislocation travels far enough away from the contact area that its strain field does not affect the measured displacement.

For the 20 nm sphere compressed at a temperature of 0 K to a displacement of 7.3 nm we find a residual plastic deformation of 5.15 nm. This was measured by comparing the position of the indenters when they first come into contact with the sphere during loading to their positions when they lose contact during unloading. Using Equation (1) with $b = 0.384$ nm and $\theta = 45°$ predicts that there should be 19 dislocations that had acted within the sphere during compression. A thorough analysis of the sphere at all imaged timesteps revealed that 24 dislocations had nucleated and traversed through the sphere to one of the sphere's surfaces, while another 2-4 dislocations were still present within the spheres after unloading. As the actual number of dislocations is greater than what is predicted, Equation 4.1 is too simple of a model for use with the geometry of the system.

However, the concept that each dislocation results in a residual displacement of roughly the same magnitude can be used to compare the results of the simulations to that of experiments. Gerberich *et al*. [4] noted that the experimental load vs. displacement plots of a 38.6 nm diameter sphere featured numerous displacement excursions with approximately the same magnitudes believed to be due to dislocation activity (Figure 4.7(a)). The total residual displacement after the first loading was seen to be close to the sum of the excursions indicating the dislocation behavior to be the primary yielding mechanism. In Figure 4.7(a) there are four clear displacement excursions and the resulting plasticity is roughly 0.9 nm. From this, each dislocation contributes an average of 0.225 nm to the total plastic displacement. In comparison, the simulation residual displacement of 5.15 nm and 24 dislocations results in a

displacement per dislocation of 0.215 nm. This shows a good agreement between the two.

It had been proposed by Gerberich, et al. [4] that the dislocations were forming as concentric loops entering into the material (Figure 4.7(b)). Given the agreement between the average displacement per dislocation seen in the simulations and experiments, it seems more likely that the dislocations would form and move in a fashion similar to what is seen in the simulations. Therefore, a better model would be of the dislocations nucleating at the contact areas on slip planes with large shear stresses and travelling out toward the far away surfaces of the sphere (Figure 4.7(c)). It should be noted that this in no way changes the results of the experimental paper [4]. In fact, the results of the simulations presented here are consistent with the claim that the primary yielding mechanism is dislocation nucleation and motion.

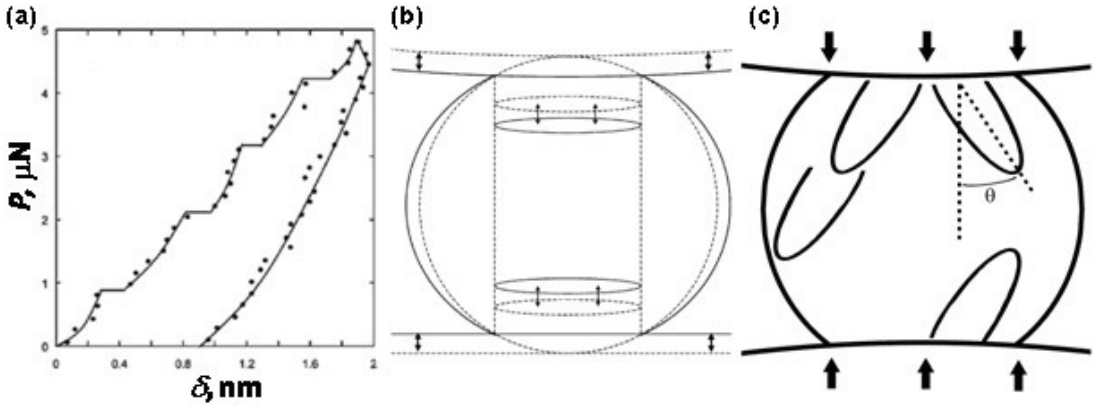

Figure 4.7: (a) The load vs. displacement curve for an experimentally compressed silicon nanoparticle showing clear excursion events (from [4]). (b) The initial model from [4] that was based upon prismatic punching of dislocation loops into the sphere. (c) This revised schematic is based on the simulation results that uses dislocation loops which nucleate at the contact area and travel at an angle through the sphere.

4.4 Hardening Behavior

The load vs. displacement curves from the simulations showed numerous drops in the applied load. Slip vector mapping of the atoms revealed that every major drop corresponded with dislocation activity (e.g. Figure 4.8). Interestingly not every dislocation event corresponded to a load drop as many of the smaller events only resulted in bumps or slope changes in the curve. Given that the simulations are displacement controlled and the experiments are load controlled, these drops in load are consistent with the experimentally observed displacement excursions.

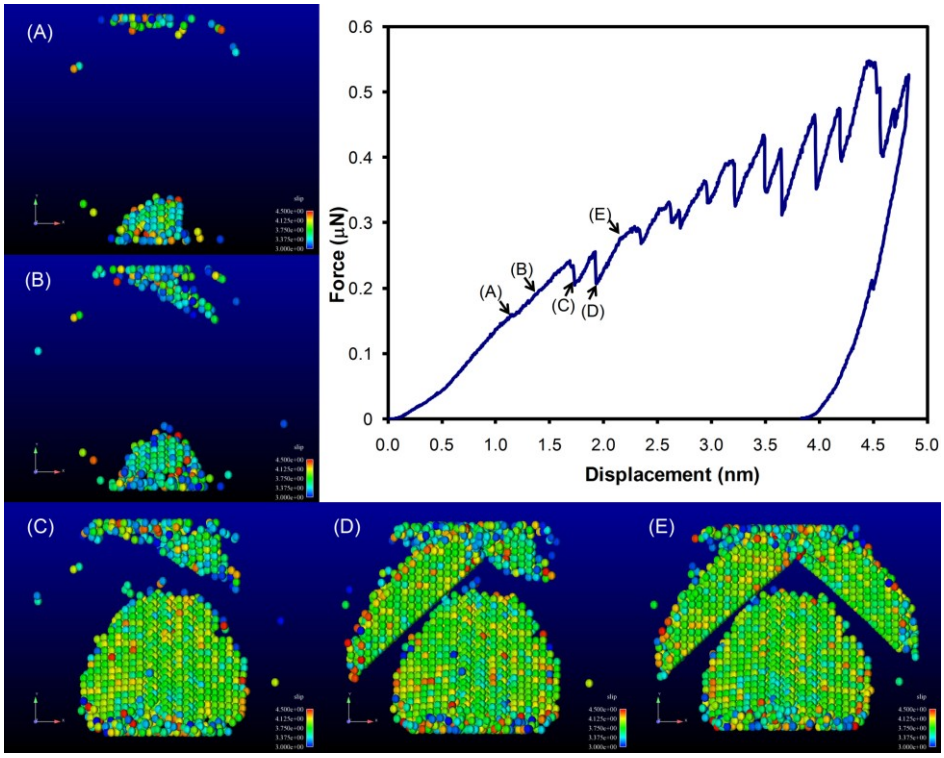

Figure 4.8: Evolution of dislocation behavior seen at moderate displacements of the 10 nm [100] compressed sphere at 300K. Images use the slip vector parameter to highlight atoms showing the path that a dislocation has traveled.

The averaged contact stress ($P/\pi a^2$) was also calculated for all of the samples with a being the radius of the contact area calculated using a formula given by Vergeles

et al. [89]. The contact stress vs. strain and the load vs. strain for the 20 nm sphere compressed at 0.01 K is shown in Figure 4.9(a). At low strains a series of peaks would appear in the contact stress values corresponding to jumps in the contact area due to surface steps on the sphere. The measured load is seen to smoothly increase during this period indicating that the material is still elastic.

Around a strain of 0.10, the stress reaches a maximum at the yield point and then shows softening of the stress as the strain continues to increases. This softening occurs as the dislocations reach the free surface and disappear. Thus hardening is impeded, as the dislocations are unable to accumulate and increase in number with increasing strain. However, for a short period of time just after the initial yielding event, this is not the case in the larger spheres.

For the 20 nm sphere at both 0 K and 300 K, the stress generally plateaus (Figures 4.9(a),4.10). During this period, the number of dislocations nucleating is greater than the number reaching the surface allowing for a short period of nearly constant hardness as the number of dislocations within the sphere increases up to a maximum of 5 (Figure 4.9). During this same strain range for the 34 nm sphere, the hardness is seen to greatly increase for a short period of time after the initial burst of eight dislocations (Figure 4.10).

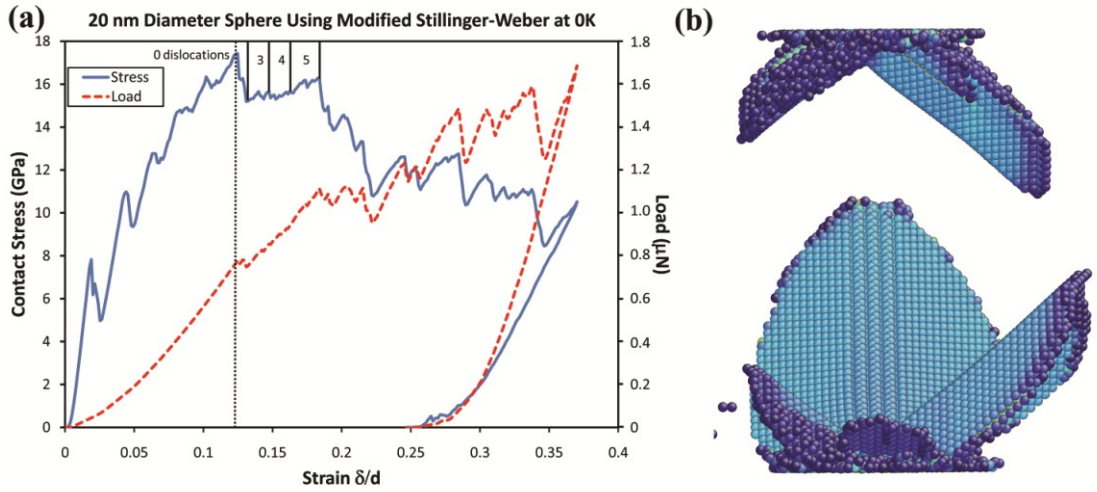

Figure 4.9: (a) The contact stress vs. displacement and the load vs. displacement curves for the 20 nm diameter sphere compressed along the [100] direction. The dashed vertical line marks the onset of yield. Solid vertical lines mark regions during the subsequent plateau behavior where the number of dislocations remains the same. (b) An image of the slipped planes within the sphere near the end of this plateau behavior.

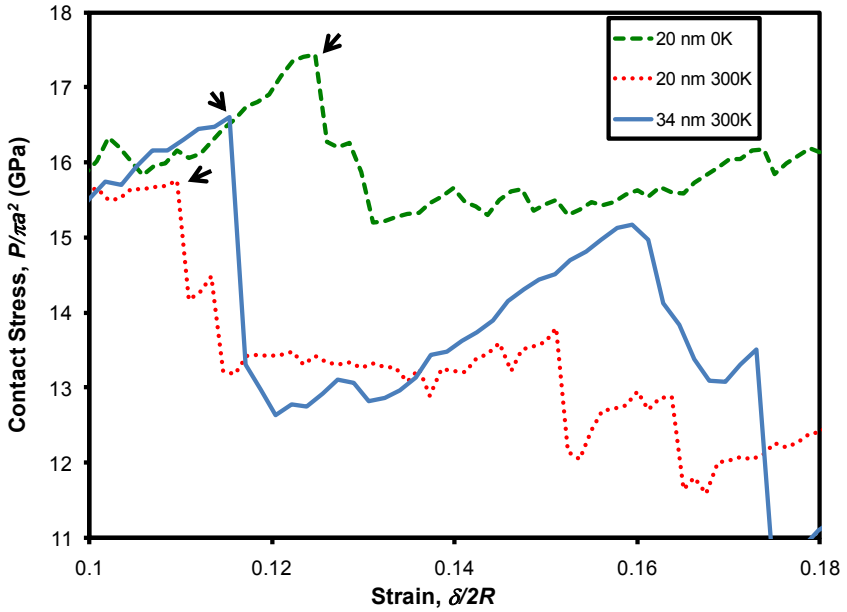

Figure 4.10: A close-up view of the stress vs. strain curves for the 20 and 34 nm compressed spheres near the first yielding. The arrows point out the first instances of plastic yielding. Note that the stress plateaus for the 20 nm data and steadily increases for the 34 nm data after the initial yield.

A method was developed to try to quantitatively compare this short-lived hardening response in the simulations to the actual hardening response from experiments. In a paper under preparation, nanospheres in the range of 40-400 nm in diameter will be shown to possess high strain hardening capacity under compression [88]. One of the spheres analyzed, 38.6 nm in diameter, also contained atoms on the order of 10^6 being comparable to the 34 nm diameter computer simulation. As described elsewhere [47], this sphere was repeatedly loaded and unloaded with a Triboscope mounted on an atomic force microscope for accurately monitoring the permanent plastic deformation before each run.

To investigate the hardening capacity, the experimental load and displacement curves for the 2nd, 5th, 7th, 9th, 12th and 15th compressions of the 38.6 nm particle were converted into stress and strain in a fashion similar to what was done with the simulations. The only complication being that the contact area cannot be directly measured for the experimental spheres. To overcome this, the Geometric contact area model [7] was used to approximate the contact area.

$$a_g^2 = R\delta_T - \frac{\delta_T^2}{4} \tag{4.2}$$

Here R is the original radius of the sphere, δ_T is the total displacement given by $\delta_T = (2R - h)$, where h is the height of the sphere before each compression. This model was chosen due to its simple form and because it has been previously used in conjunction with experimental results [7, 47]. Whatever the combination of elastic and plastic strain, this Geometric relationship represents the total contact area assuming that

there is not a large barreling of the sphere. As the strains are relatively small, this is a reasonable assumption.

Plotting the stress vs. strain data for both the experiments and simulations on a log-log plot (shown in Figure 4.11) revealed that they all featured power law regions that started around 10% strain. A least squares fit was done on these regions to calculate a hardening exponent. While this method may not produce a true hardening exponent as contact stresses were used rather than mean stresses, it nevertheless allows for a quantifiable method of comparison.

For the 38.6 nm data, the 2nd, 5th, 7th, 9th, 12th and 15th compressions resulted in hardening exponents of 0.17, 0.77, 0.81, 1.17, 2.25 and 1.99 respectively. This indicates a trend in which the hardening behavior becomes more pronounced as the sample is repeatedly loaded suggesting that plastic damage is rapidly accumulating.

In comparison, if the same fitting was made for the whole plastic region of the simulations, the hardening exponents found range from -0.3 to -0.6 indicating extensive softening. As previously mentioned, this softening most likely results in the dislocations that form not being constrained within the spheres. Thus to compare the hardening of the simulations with the experimental work, it makes sense to only examine the hardening behavior in the region just after the initial yielding where dislocations are present within the sphere but haven't had the chance to escape to the sphere's surface. By looking at these regions only, hardening exponents for the 0.01 K and 300 K compressions of the 20 nm sphere are found to be 0.18 and 0.04 respectively while the 34 nm simulation shows a hardening exponent of 0.64.

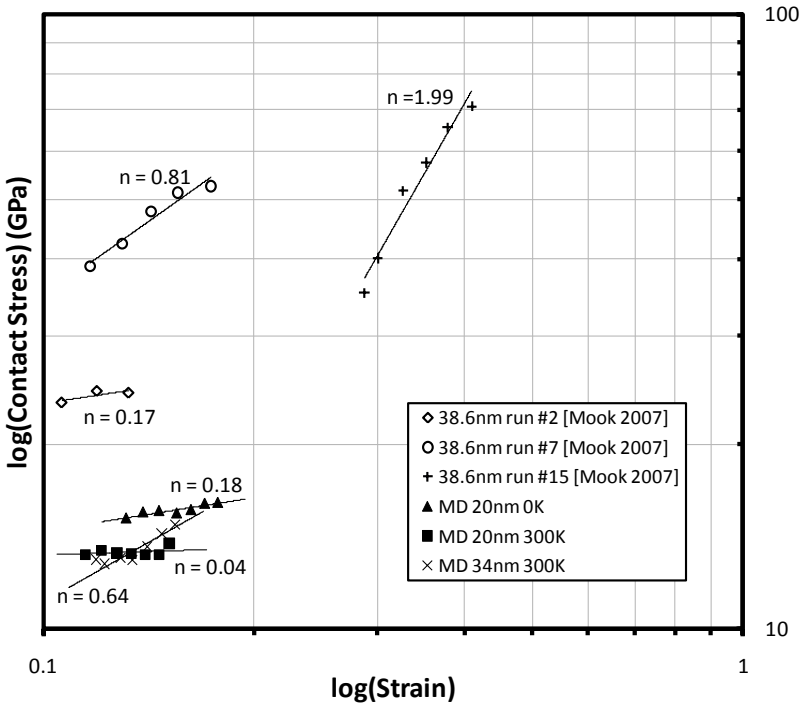

Figure 4.11: The strain hardening curves obtained from the simulations and three of the 38.6 nm compression runs.

Since the simulated spheres are only loaded once, it would be expected that the measured hardening exponents should be consistent with the earliest compressions of the experimental particle. For the 20 nm sphere, this is indeed the case with the 0.01 K run having a hardening exponent comparable to the second experimental loading. There is, however, a discrepancy between the magnitude of the contact stress values measured as even the second experimental loading shows values greater than all of the simulations.

The exponent for the 34 nm run is larger and more consistent with the 7th experimental loading of the 38.6 nm sphere, i.e. n=0.64 and 0.81 respectively. The higher hardening exponent measured for the 34 nm sphere could be attributed to a number of different reasons. First, since more dislocations are observed to be present

within this sphere after the initial yield, the back stresses associated with those dislocations cause a sharper increase in the measured hardness. Since the simulations used here were selected to show the most dislocations they may show more dislocations than what occurs experimentally resulting in a larger hardening exponent.

Another possibility is that the large hardening exponent observed in simulations could result from partially elastic behavior. The first instance of yield in this sphere is the formation of 8 dislocations within a short time period. This initial burst of dislocations may act as a single yield event after which the sphere may respond elastically for a short period until another event occurs. However, the slope of this region $d\sigma/d\varepsilon$ is about 70 GPa, which is less than half the elastic modulus of silicon indicating that it is not entirely elastic. This could be clarified with simulations by obtaining more data from more runs or being able to measure the post-yield behavior for a longer period without the dislocations reaching the free surface and disappearing.

4.5 Conclusions

Molecular dynamics simulations using the Stillinger-Weber potential of compressed silicon nanospheres reveal extensive dislocation and dislocation related yielding behaviors. Depending on the orientation, dislocations are seen to nucleate near the contact points and grow on both {110} and {111} type planes. In addition, stacking faults and regions of BCT5 are also seen to occur. The extensive presence of the dislocations within the sphere sizes simulated here show that the Stillinger-Weber potential offers the opportunity to efficiently study dislocations in nanostructures of silicon with MD.

The unexpected presence of dislocations on {110} planes is investigated. By estimating the energy barrier for dislocation nucleation with generalized stacking fault calculations, it was shown that for the Stillinger-Weber potential, the barrier on {110} planes is comparable to the barrier on {111} shuffle-set planes. The low energy of the {110} slip was attributed to the presence of a metastable state along the slip pathway in which the atoms form BCT5-like bonding. A comparative study using DFTB simulations showed the barrier of the {110} slip to be considerably larger than that of the shuffle-set.

Molecular dynamics simulations of silicon nanosphere deformation are presented. These make a clear case that dislocation formation and interaction are the primary mechanisms behind the measured behaviors of silicon nanospheres under confined compression. The simulations show a direct correspondence between dislocation nucleation and drops in the measured load. This confirms that the displacement discursions that are observed experimentally for these compressions of small spheres are most likely due to dislocations.

The measured contact stress of the simulated spheres is also seen to increase with strain during periods where the number of dislocations present is increasing. A hardening exponent was measured during this region, for which the 20 nm spheres showed hardening consistent with experimental results. However the 34 nm sphere indicated a much larger hardening exponent than what is consistent with experiments for spheres that had been compressed only once. This high hardening could either be due to more dislocations present within the sphere than what occurs experimentally, or from the multiple dislocations acting together as discrete yielding events. All of the

results show that the presence of the dislocations within the spheres corresponds to a region of increasing hardness similar to or greater than what is seen experimentally. This indicates that the experimentally observed hardening at low strains is consistent with dislocations forming within the nanoparticles.

Chapter Five: Dislocation interactions

The focus of this chapter is a bit different than the previous two. The previous chapters focused on using molecular dynamics to compare simulated behaviors to experimental results from similar conditions. In contrast, this chapter compares results from simulations to theoretical models involving dislocation interactions. The purpose of this study is to obtain a better understanding of the dislocation interactions and determine if simulations can be designed to appropriately investigate those interactions.

In order to study the dislocation interactions, various external constraints were imposed upon simple simulation systems. It was observed that the constraints greatly influenced the nature of the dislocations within these simulations allowing for a number of different interactions to be studied. Specifically, the direct influence of the constraints on the dislocations were observed, along with dislocation-dislocation interactions and dislocation-interface interactions. The interaction between dislocations and free surfaces are compared to how the dislocations respond to the presence of an oxide layer.

5.1 Si-SiO_2 Potentials

Prior to any investigations of how dislocations within silicon interact with an oxide layer, an appropriate potential has to be found that can adequately handle the bonding and behavior of the pure silicon, the oxide and the interface between the two. Two such Si-SiO_2 potentials were investigated here: the potentials of Jiang and Brown

[72, 73] and Watanabe, Fujiwara, Noguchi, Hoshino and Ohdomari [76, 77]. These two potentials are similar in that they both modify the Stillinger-Weber potential to allow for oxygen atoms to be properly introduced into the system.

In order to study these potentials, they both had to be implemented into the code of the LAMMPS molecular dynamics program [83]. The files developed for the implementation of these potentials are included in Appendix Two. As the potentials were being implemented, various simple test runs were performed to compare the results with what was previously reported with the potentials to insure that the potentials were working properly. During this preliminary testing, it was determined that the Watanabe potential could adequately simulate an oxide layer on a silicon crystal, while the Jiang and Brown potential was inadequate for this.

The Jiang and Brown potential was originally created to simulate the addition of oxygen impurities into bulk silicon. As such, the results of the simulations from the first papers using this potential reported values related to this impurity [72, 73]. In particular, the resulting dilation of the crystal and enthalpy of formation, ΔH_f, related to the addition of a sole oxygen atom were reported, along with the equilibrium Si-O bond length.

During the preliminary testing of the Jiang and Brown implementation in LAMMPS, a discrepancy was observed. To handle the oxide, the Jiang and Brown potential mixes the Stillinger-Weber silicon potential with the BKS SiO_2 potential. Present within the BKS pair term is an expression for the Coulombic interaction of Si and O ions

$$\phi_{BKS}^{Ionic} = \frac{Cq_i q_j}{r_{ij}} \qquad (5.1)$$

where q_i and q_j are the ionic charges related to ions i and j and C is Coulomb's constant. In order to obtain similar values with the potential for the dilation, ΔH_f, and Si-O bond length, Coulomb's constant needed to be neglected. The values associated with these three values found here by including and not including C are plotted in Figure 5.1 along with the reported values.

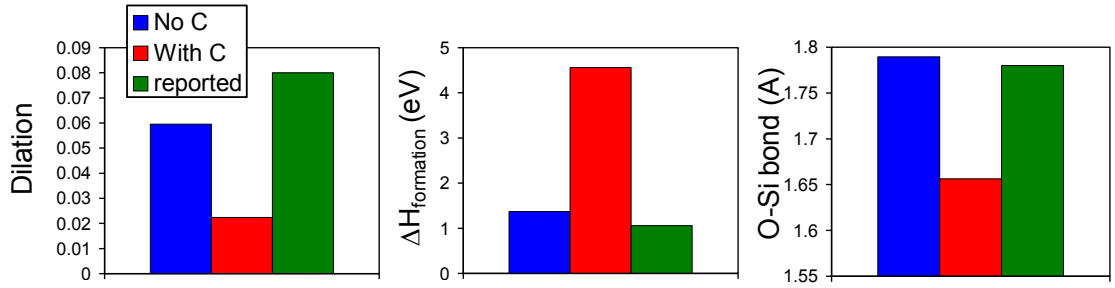

Figure 5.1: Various values obtained with the Jiang and Brown potential reveal that Coulomb's constant, *C*, must be neglected in order to match with the reported values [72, 73].

Leaving out the Coulomb's constant did allow for these properties to be decently matched with the reported values, but it had drastic effects on the modeling of bulk regions of SiO_2. A standard of an ideal crystal of α-quartz was used to compare the oxide found by the BKS potential [74, 75] with the oxide calculated with the Jiang and Brown potential. Computing the radial distribution function related to the first peak revealed that Coulomb's constant must be included in the Jiang and Brown potential in order to properly model the bonds within the oxide (Figure 5.2). The simulations to obtain the RDF curve also revealed that the oxide became unstable after a period of time with the Jiang and Brown potential regardless of the inclusion of the Coulomb's

constant. This clearly indicates that the Jiang and Brown potential is inadequate for modeling a layer of oxide on a crystal of silicon.

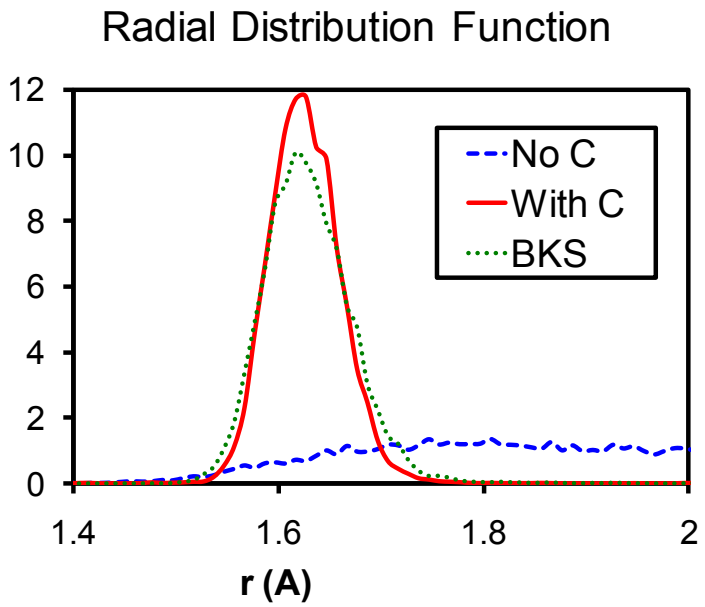

Figure 5.2: The radial distribution function for the first peak of α-quartz using the Jiang and Brown potential. A Comparison with the BKS oxide potential is shown indicating that Coulomb's constant should be included to properly model the oxide.

Similar preliminary tests with the Watanabe potential revealed values consistent with those reported. As the Watanabe potential is a modification of the Stillinger-Weber potential, it is expected that regions of oxide free silicon should behave identically in both the Watanabe and the Stillinger-Weber potential given that the same parameters are used for the Si-Si interaction. A test simulation of a bulk crystal of silicon revealed that the Watanabe potential did produce the same results as the Stillinger-Weber potential, albeit a slightly longer computation time. Further tests investigating the Si-O bond length for a two atom system and within α-quartz, and the potential energy of formation for the α-quartz crystal matched with what was reported.

The Watanabe potential has been previously used to study Si-SiO_2 systems showing that both phases and the interface between the two are stable [76, 77, 81, 82]. Also, the behavior of the pure silicon is identical to that given by the Stillinger-Weber potential making the nature of the dislocations consistent between the oxide containing systems and the oxide free systems. Finally, the earlier works have shown that the oxide can be successfully grown using the routine of Dalla Torre, et al. [82] that is described in Chapter 2. Because of all of this, the Watanabe potential was chosen for the study of the dislocation interactions with the oxide layer.

5.2 System Design

While the system design is explained in depth in Chapter 2, a quick description is presented here due to it being distinctly different than the spherical particles explored in Chapters 3 and 4, and the fact that the results depend heavily upon the specific conditions implemented. The systems studied here consisted of rectangular blocks of atoms in initially perfect diamond cubic silicon positions. While a variety of simulation sizes were explored, they all were designed to be similar. All the simulations had periodic conditions in the z direction with box dimension in that direction of 26.606 Å. The x and y directions were held proportional to 85.846 Å by 94.068 Å, with simulation sizes used that were 2X, 3X, 4X and 5X times this in both of these directions. Each simulation contained two perfect edge dislocations created by removing planes of atoms. These dislocations were initially positioned at 1/2 the x dimension and at 1/4 and 3/4 the y dimension.

The results presented here are divided up based upon the boundary conditions in the x direction. This was done as the results and what interactions that could be studied depended highly upon the boundary conditions. The three types of boundary conditions investigated were periodic x boundaries, free surfaces consisting of non-periodic, non-constrained surfaces, and oxide surfaces created by adding in a region of oxide. The periodic boundaries allowed for the study of how the dislocations interacted with each other and with strains imposed upon the simulation box. Incorporating the free surface or oxide surface allowed for a comparison of how the dislocations interacted between the two simulated surfaces.

In addition to investigating different boundary conditions in the x direction, the conditions imposed upon the y boundaries also influenced the results. Two conditions were used in particular: constraining the atoms near the surface to be held in place, and leaving the boundary atoms unconstrained resulting in a free surface. Constraining the atoms at the y-boundaries fixes the strain within the system. This affects the movement of the dislocations because a moving dislocation results in a small displacement between two planes of atoms which in turn causes a strain within the system. In contrast, leaving the y-boundaries free allows for the dislocations to respond to the local internal interactions.

5.3 Periodic Boundary Conditions

Periodic boundary conditions in the x direction offer the simplest situation for studying. The only interactions that are present are interactions between the dislocations, the strain within the whole system and possibly some image forces from

the surfaces in the y direction. How the y boundaries are treated affects which of the interactions is dominant. With unconstrained y-boundaries the two dislocations move away from each other due to the interaction between the two dislocations. According to elementary dislocation theory, the interaction force between these two parallel edge dislocations is repulsive when the separation distance between the two dislocations in the x direction is less than their separation in the y direction, and attractive when the x separation is greater than the y separation [93]. So for an isotropic material, if only two dislocations are interacting, they would reach an equilibrium point when their x and y separations are equivalent.

For these simulations, in addition to the system not being isotropic, the periodic condition effectively makes it so that there are not just two dislocations interacting but two infinite arrays of equally spaced dislocations. This results in the equilibrium position of the dislocations occurring when the dislocations are separated along the x direction by approximately half the periodic distance. At this position, each dislocation is equidistant from two occurrences of the other dislocation. An example for the 4X simulation is that the y separation is 186 Å, but the x separation fluctuates around 152 Å, which corresponds to half of the box distance in the x direction.

Constraining the atoms at the y boundaries results in a different dislocation response. By holding the y boundaries in place, the total strain of the system is held at zero. This restricts the movement of the dislocations by making it unfavorable for the dislocations to separate. In the end, the dislocations are seen to move apart slightly due to their repulsion, but not by much due to the system strain restriction. For that same 4X simulation mentioned before, the x separation of the dislocation only averages around

8.5 Å as opposed to the 152 Å seen with the unconstrained conditions. The separation distance is plotted vs. the simulation run time in Figure 5.3.

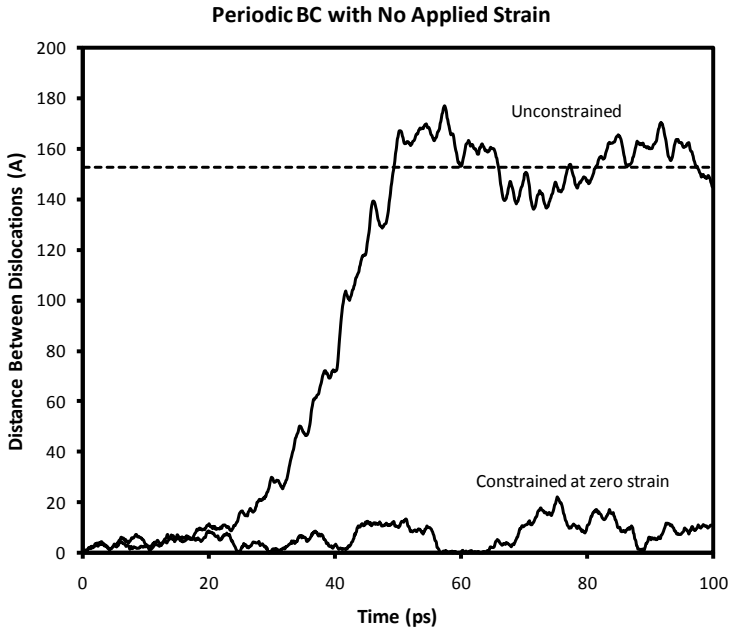

Figure 5.3: The separation distance between the two dislocations in the 4X periodic simulation. When unconstrained, the dislocations repulse each other and separate by exactly half the periodic distance. Constraining the system to a zero strain state prevents the dislocations from separating.

The effect of the constrained boundary on the dislocation behavior can be calculated from geometrical considerations. Within the periodic simulations, each complete cycle of a dislocation back to its original position would displace the areas above and below the dislocation by exactly one Burgers vector, *b*. So, one dislocation traveling one full cycle (a distance of $\Delta x = x_{hi} - x_{lo}$) corresponds to a total geometric shear displacement, δ, of *b*. The resulting displacement due to one dislocation moving an arbitrary amount is therefore given by

$$\delta_x^{\perp} = b\frac{x^{\perp}(t) - x^{\perp}(0)}{\Delta x} \tag{5.2}$$

where $x^{\perp}(0)$ is the initial x coordinate of the dislocation and $x^{\perp}(t)$ is the x coordinate at time *t*. All of the systems studied here have two dislocations of opposite signs that initially have the same x coordinate. This results in the total displacement caused by both dislocations to be given by

$$\delta_x^1 + \delta_x^2 = b\frac{x_1^{\perp}(t) - x_1^{\perp}(0)}{\Delta x} - b\frac{x_2^{\perp}(t) - x_2^{\perp}(0)}{\Delta x} = b\frac{x_1^{\perp}(t) - x_2^{\perp}(t)}{\Delta x} \tag{5.3}$$

The displacement due to the dislocations is related to the distance that they move apart from each other. In addition to the motion of the dislocations, displacements can also be imposed upon the system by moving the constrained y boundaries in the x direction. Either of these displacements result in a shear strain within the system given by

$$\gamma_{xy} = \frac{\delta_x}{\Delta y} = \frac{\delta_x^{\text{applied}}}{\Delta \text{y}} + \frac{b\left(x_1^{\perp} - x_2^{\perp}\right)}{\Delta x \Delta y} \tag{5.4}$$

This in turn results in a force acting upon the dislocations from the strain within the system as

$$F_x^{\text{box}} = \mu b \gamma_{xy} = \frac{\mu b \delta_x^{\text{applied}}}{\Delta \text{y}} + \frac{\mu b^2\left(x_1^{\perp} - x_2^{\perp}\right)}{\Delta x \Delta y} \tag{5.5}$$

Within these constrained systems, this force acts upon the dislocations by attempting to relieve the shear strain. Neglecting all other forces acting on the dislocations shows that for an applied displacement of the box the equilibrium separation is

$$\left(x_1^{\perp} - x_2^{\perp}\right) = \frac{\Delta x \delta_x^{\text{applied}}}{b} \tag{5.6}$$

If the constrained atoms are moved at a constant rate, then the average of the dislocation velocities would have to reach a steady state in order to compensate for it. The average steady state velocity is

$$\bar{v}_{\perp} = \frac{\Delta x \dot{\delta}}{2b} \tag{5.7}$$

where $\dot{\delta}$ is the displacement rate $d\delta/dt$. Keep in mind that the dislocations are traveling in opposite directions as they have opposite signs.

Measuring the velocities of the dislocations within periodic systems subjected to a constant displacement rate reveals that this model is accurate. Plots of the dislocation positions vs. time revealed that the dislocations moved at nearly constant rates after strain was applied. This allowed for velocities to be measured using a linear least squares fit to the post strain region. The velocities were measured within all of the sizes investigated and at a variety of displacement rates. It was found that at the slower velocities, the measured velocities are consistent with and only slightly less than the predicted steady state values of Equation 5.7. This slight difference is most likely due to the time scale of the simulations having not quite reached the steady state. This also explains why the larger simulation sizes show slightly lower velocities than the smaller simulations at similar predicted velocities as the larger simulations would take longer to reach the steady state. At the faster displacement rates, a deviation is seen where the measured velocities appear to reach an upper limit around 29 Å/ps. This limit would be roughly 2900 m/s, which is on the order of the sound velocity of silicon (the longitudinal and shear velocities along the [100] direction are approximately 8500 and 5000 m/s respectively with the Stillinger-Weber potential).

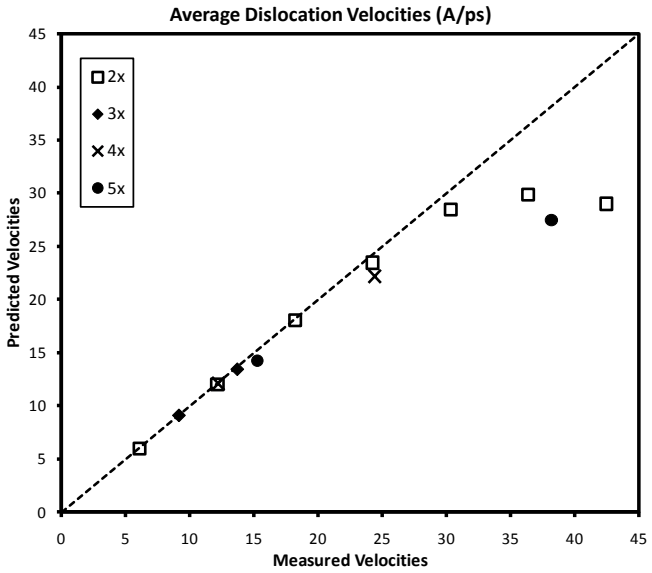

Figure 5.4: The average measured velocities of the dislocations plotted vs. the predicted steady state velocities (Equation 5.7) for different simulation sizes. An excellent agreement is seen between the measured and predicted velocities at the low strain rates. At the higher rates, the dislocations reach a maximum velocity around 29 Å/ps.

5.4 Free Surface Boundaries

5.4.1 Attraction to the Free Surface

Introducing free surfaces into the system adds additional interactions to the dislocations. Dislocations are attracted to a free surface due to the modification in the stress field that is necessary due to the constraints associated with the free surface. This results in what is referred to as an "image force", where the attraction is modeled as the presence of a dislocation of opposite sign equidistant from the free surface being placed in the "void" [93]. For an edge dislocation in an isotropic material, this results in a force of

$$F_X^{\mathrm{Im}} = -\frac{\mu b^2}{4\pi(1-\nu)d} \tag{5.8}$$

where d is the distance that the dislocation is from the free surface. Using Newton's Law

$$F_X^{\mathrm{Im}} = m_\perp a = -\frac{\mu b^2}{4\pi(1-\nu)d} \tag{5.9}$$

If the mass associated with the dislocation is constant, then it can be easily seen that

$$a \propto -d^{-1} \tag{5.10}$$

With free y boundaries, only repulsion between the two actual dislocations and attraction to the free surfaces is expected and observed. The dislocations first fluctuate about their initial positions near the center of the system approximately equidistant from the two free x surfaces. This only lasts for a short period before the dislocations start increasing in velocity as they move toward opposite free surfaces. The velocity of the dislocations is observed to continue increasing until they reach the surface and form an atomic surface step.

A simple model was used to numerically integrate the position and velocity of the dislocation as a function of time using the Velocity Verlet routine [64]. Using the interaction acceleration of $a \propto -d^{-1}$ and ignoring the dislocation-dislocation interactions, the only necessary parameters are a starting position and velocity of the dislocations and the scaling constant for the interaction. Figure 5.5 shows the positions of the dislocations as they approach the free surfaces and the corresponding simple model described. It can be seen that the model does describe the general behavior of the dislocation's positions. The model is not perfect because it neglects the dislocation-dislocation interaction and does not account for the thermal fluctuations inherent in these simulations.

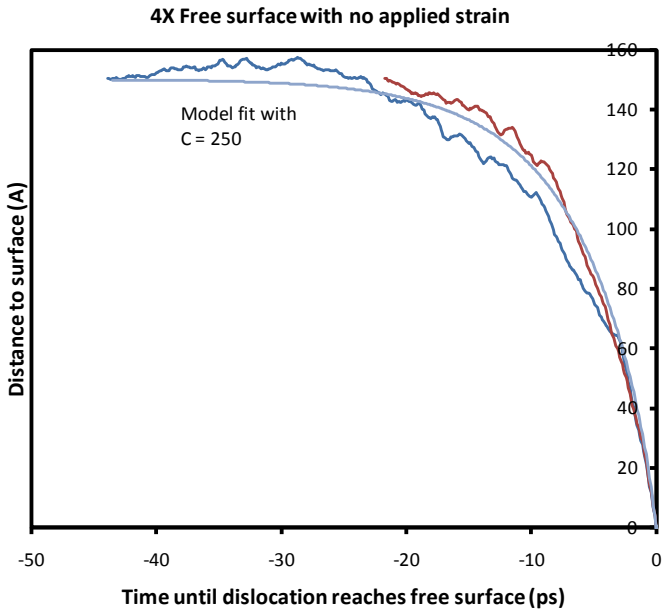

Figure 5.5: The position of the dislocations with respect to the free surface for the unconstrained 4X simulation. The acceleration of the dislocations is consistent with the model based upon a 1/d attraction.

Obtaining the fitting constant for the interactions by visually matching the model to the measured positions indicates that it might depend on the size of the simulation. For instance, the acceleration constants found for the 2X, 3X, and 4X simulations were 350, 300 and 250 $(Å/ps)^2$ respectively. This slight size dependence could be due to the neglected dislocation-dislocation interactions which would be greater for the smaller simulations as the dislocations are closer to each other and therefore causing a greater repulsion towards the free surfaces.

With constrained and moving boundaries, the interaction with the free surface can be overpowered by the interaction resulting in behaviors that are a combination of the interactions between the dislocations, with the free surfaces and with the system strain. Having the free surfaces constrained but not moving, the two dislocations

eventually reach the same free surface. In the 2X and 3X systems, the two dislocations stay close to each other and move towards the same free surface. In contrast, a more complex response is seen for the 4X and 5X runs. Initially, the dislocations repulse each other just like with the periodic boundaries. After separating, the dislocations were then closer to the free surfaces resulting in one of the two dislocations being pulled to a free surface and forming a surface step. Immediately after, the other dislocation then moved away from the free surface that it was closest to and eventually reached the same surface that the first dislocation had disappeared on. This reversal of the second dislocation's movement occurs as the system strain due to the two dislocations separating still remains in the system, but the dislocation-dislocation repulsion is gone as there is now only one dislocation. The system strain overpowers the attraction of the closer free surface and moves the dislocation to the free surface that the other dislocation had disappeared at. Figure 5.6 illustrates this response.

Constant strain rate simulations with the free surface initially show dislocation positions and velocities similar to what was seen during the comparable periodic simulations. However, after a short time, the velocities of the dislocations increase indicating an attraction towards the free surfaces.

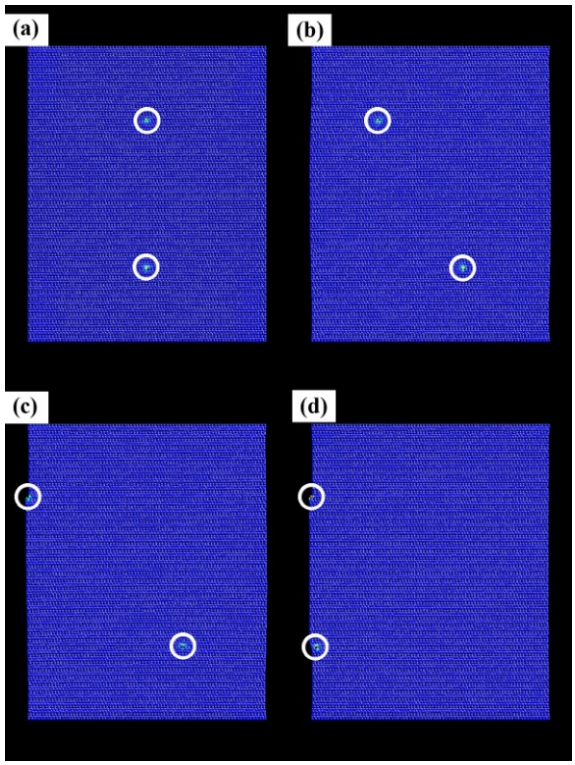

Figure 5.6: The dislocation response seen for the 4X simulation with free x surfaces and y surfaces constrained at zero shear strain. The atoms are colored according to their potential energy. The positions of the dislocations are highlighted with white circles for clarity. (a) Initial positions of the dislocations. (b) The dislocations repulse each other. (c) The top dislocation is attracted to a free surface. (d) The bottom dislocation reverses direction and goes to the farther free surface because of the strain within the system.

5.4.2 Temperature Variations

The free surface simulations were also used to study the motion of the dislocations at various temperatures. As the attraction of the dislocations to the free surfaces is the dominant force and has a consistent behavior, it was used as a standard for investigating the dislocation motion. The free surface attraction also had the benefit of developing naturally without any artificial constraints placed upon the system box.

Using the 2X simulation size, the motions of the dislocations in an unconstrained system were observed for temperatures between 0.01 K and 300 K. The motion of the dislocations towards the free surface was practically indistinguishable for all of the simulations that were run between 10 and 300 K. Below 10 K, the dislocations are observed to no longer smoothly approach the surface, but instead show a stepwise behavior where they remain at one spot for a short period of time before jumping slightly closer to the free surface (Figure 5.7). Progressively lower temperatures show this effect to become more pronounced and the total time for the dislocations to reach the surface increases. At 0.01 K, the dislocations only slightly move from their initial positions and remain fixed for the remainder of the simulation never reaching the surface.

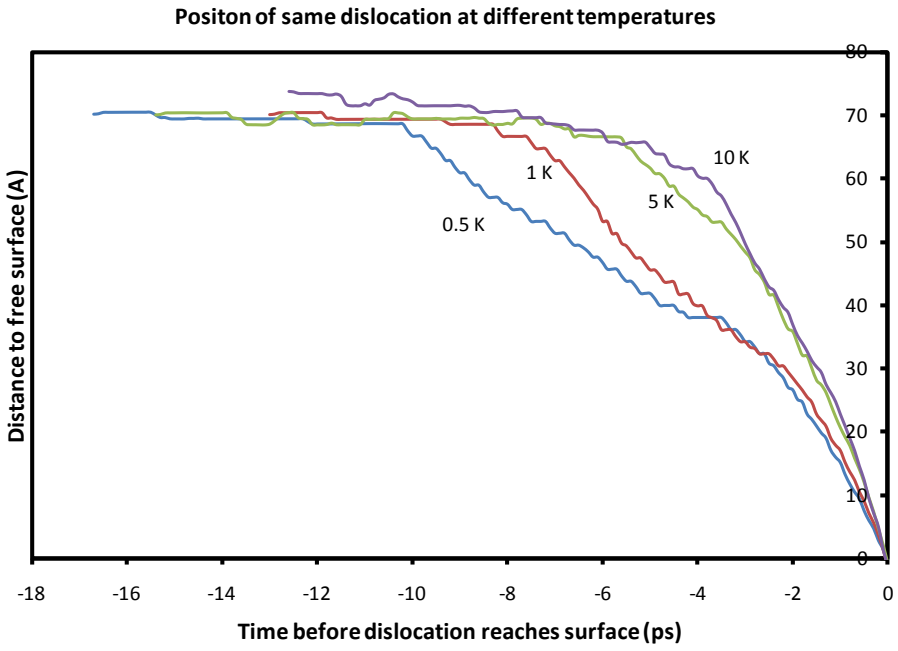

Figure 5.7: The position of one dislocation as it is attracted to a free surface at various low temperatures. As the temperature decreases, the time it takes for the dislocation to reach the surface increases and a stepwise motion becomes evident.

The variation in the dislocation motion with temperature can be explained with the Peierls energy barrier. As the dislocations move through the crystal, there are certain positions that are lower in energy due to the resulting atomic configurations. The dislocations jump from one low energy position to the next when they have enough energy to overcome the barrier associated with crossing the higher energy positions. Here, the dislocations are moving along the [110] direction. The average distance measured between the jumps is close to the (440) planar separation within DC silicon of 0.96 Å.

The energy needed to overcome the barrier has to either come from the forces acting on the dislocations or from the thermal energy of the system. For the low temperature simulations, the force acting upon the dislocations is not large enough to overcome the Peierls barrier when the dislocations are near their initial positions. Thus, the dislocations require thermal energy from the system in order to move. As the temperature decreases, it becomes less likely for the atoms making up the dislocation core to have enough kinetic energy to allow the dislocation to change positions, so the dislocations remain at the low energy positions for a longer time.

Also, as a dislocation gets closer to a free surface, the attractive force acting on the dislocation increases making it more likely for it to move resulting in shorter times before the next jump in position. Eventually, the attractive force gets strong enough that the low temperature no longer matters and the dislocation begins to move in a smoother fashion.

This technique, or a similar one, could conceivably be used to work out the activation energy associated with the Peierls barrier and how it depends both on

temperature and the forces acting on the dislocation. However, a much more robust model of the dislocation attraction is needed. While the acceleration due to the attraction was shown to be consistent with the models, the actual expression for the force was not identified as the effective mass of the dislocations is unknown and the models used were based upon an isotropic medium. Working out an exact expression for the attraction would require either a more accurate model based upon anisotropic elasticity, or a method of identifying the dislocation's effective mass through a comprehensive simulation study. Either of these seem possible for continuing this work.

5.5 Oxide Interaction

Three different system configurations were investigated for studying the interaction of the dislocations with the oxide interface. All of these systems were created by attaching atomic blocks containing an oxide onto the dislocation containing systems used during the periodic and free surface simulations. The growth of the oxide containing blocks is outlined in Chapter 2. The first system configuration featured an oxide layer added to one of the x surfaces by matching up the atoms at the x interface that were fixed at perfect DC positions with the fixed atoms at the base of the block upon which the oxide was grown. This allowed for the oxide to be introduced to the new system without placing any atoms in high energy configurations, but resulted in the distance between the dislocations and the oxide interface being greater than the distance between the dislocations and the free surface at the opposite x surface. The second configuration was created in a similar manner, but an oxide was added to both x

surfaces. Finally, the third system that was investigated featured oxide interfaces that were placed at positions comparable to the positions of the surfaces in the free surface systems. However, this required the two base systems to be merged between atoms that were already relaxed from ideal positions creating a temporary higher energy region.

Assuming isotropic behavior and a smooth interface, the interaction between a dislocation and an interface with a different material can also be calculated with image forces. In this case, the dislocation would still feel a force as if there was an imaginary dislocation equidistant on the other side of the interface. However, now the Burgers vector associated with that imaginary dislocation, b^*, is related to the shear moduli of the two materials that form the interface [93]

$$b^* = b\frac{\mu^* - \mu}{\mu^* + \mu} \tag{5.11}$$

where μ is the shear modulus of the material containing the actual dislocation and μ^* is the shear modulus of the material on the other side of the interface. Using this, we find that the force on the actual dislocation is

$$F_X^{Int} = \frac{\mu b b^*}{4\pi(1-\nu)d} = \frac{\mu - \mu^*}{\mu^* + \mu} F_x^{\mathrm{Im}} \tag{5.12}$$

Using some generic values related to silicon and fused silica, this interaction is estimated to be roughly 1/4 to 1/3 of the interaction with the free surface. This predicts that the interaction should have a similar 1/*d* dependence as the free surface, but that the strength of the oxide interaction is smaller.

5.5.1 One Oxide Interface

For the simulations with one oxide interface and one free surface, it is difficult to work out the oxide interaction within these simulations as it is never the dominant interaction. Initially, the attraction to the free surface is dominant as noted by the dislocations moving towards it. This is to be expected as the attraction to the free surface is predicted to be stronger than the attraction to the oxide interface and the dislocations are initially closer to the free surface than the oxide interface. If no shear strain is applied, or if the shear is too small to counter the free surface attraction, both dislocations will reach the free surface.

With an adequately large applied strain rate, the attraction to the free surface can be countered for one of the dislocations and continued straining moves that dislocation towards the oxide interface. As the shear increases, it quickly becomes the dominant interaction acting upon the dislocation. During this stage, the velocity of the dislocation should approach twice the steady state velocity given by Equation 5.7 as there is only one dislocation still active within the system. A plot of the dislocation's position with time shows that this is the case for the smallest simulation (Figure 5.8). For the larger simulations, the dislocation velocity is smaller than the predicted steady state velocity.

When the dislocation is within 10-20 Å of the oxide interface, it is observed to move faster for all of the simulation sizes. In the smallest simulation, the dislocation moves towards the interface quicker than what the steady state velocity predicts indicating an attractive effect. For the larger simulations, the velocity during this range is seen to be comparable to the steady state value.

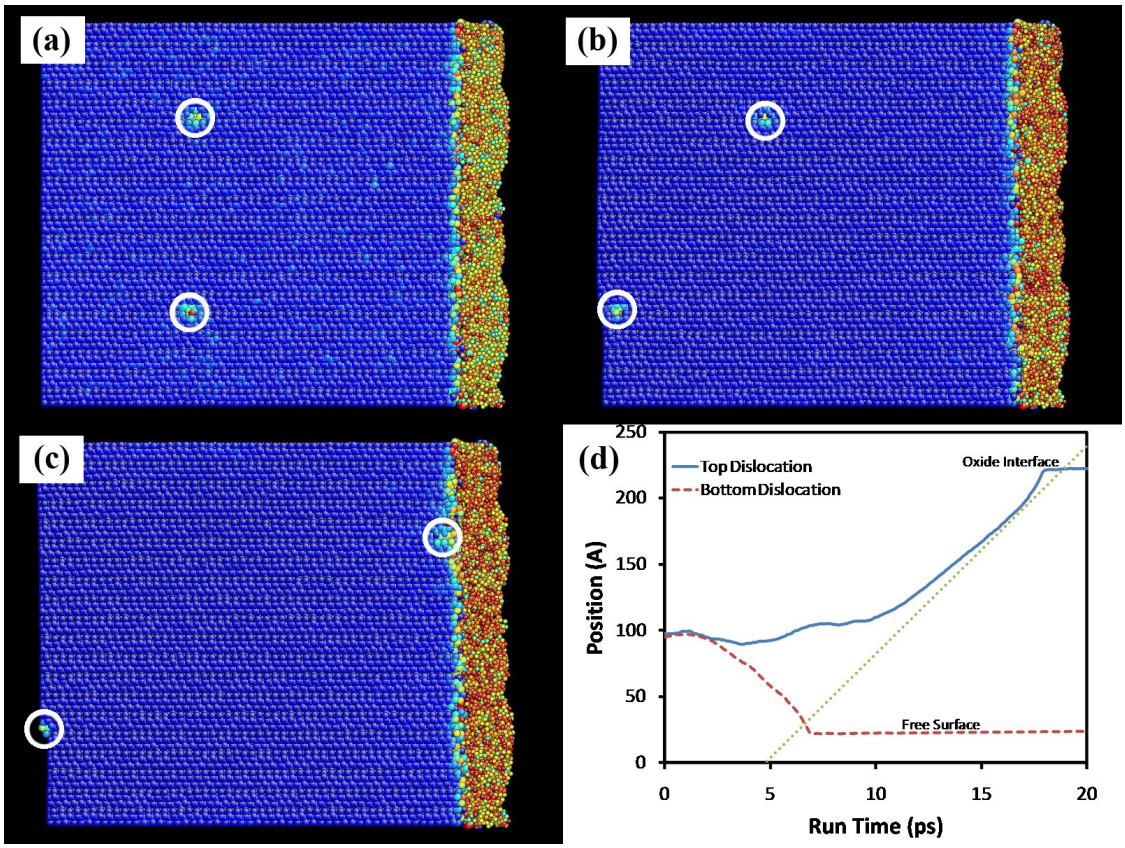

Figure 5.8: The position of the dislocations in the one oxide system as a constant shear displacement is applied. The top dislocation is seen to approach the predicted steady state velocity then accelerate faster when close to the oxide interface.

5.5.2 Two Oxide Interfaces

The second and third system designs that placed oxide layers on both x surfaces eliminated the issue of the free surface attraction overpowering any interaction with the oxide interface. This allowed for the long range interaction of the dislocations with the oxide interfaces to be studied.

Applying a constant shear displacement causes the two dislocations to move away from each other and towards opposite oxide interfaces. Similar to what was observed with the single oxide system, the velocity of the dislocations in the smallest

simulation are comparable to the steady state value obtained from Equation 5.7, while the larger simulations show the measured velocities to be smaller than the steady state. Also, the velocities appear to increase when the dislocations are within 10-20 Å from the interface.

The position vs. time plots for the larger simulations (Figure 5.9(b)) did reveal some interesting behaviors. Although the dislocation velocities are not at the predicted steady state values, they nevertheless remain constant for a period of time. Then, nearly simultaneously, the two dislocations increase speed and remain at the new rate until the dislocations are within 10-20 Å from the interface, at which point they increase speed again. This stepwise increase in velocity is intriguing as it suggests a sudden change in the interactions with the dislocation, either due to the oxide or the simulation design.

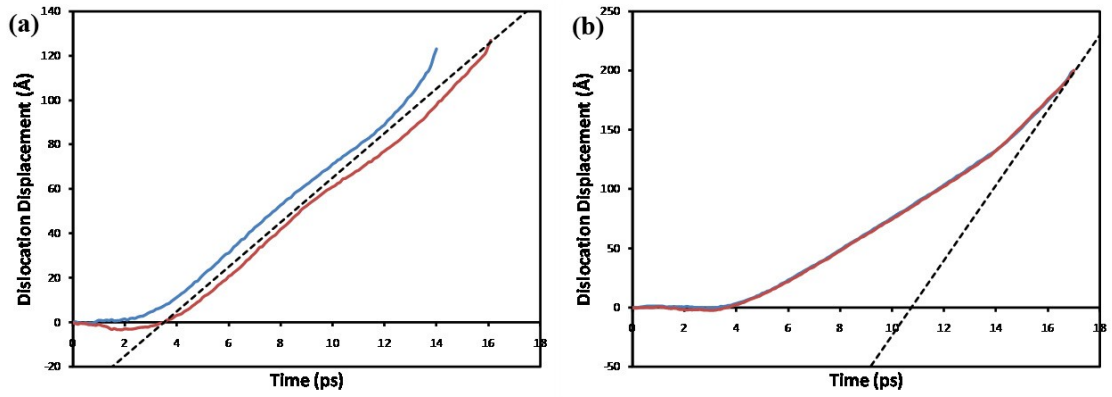

Figure 5.9: The displacement of the dislocations with time due to a constant applied shear strain for the systems with oxide layers on both x surfaces. (a) The dislocation response for the 2X simulation matches with the steady state behavior (dashed line) before accelerating near the interface. (b) The dislocation response for the 4X simulation showing regions of consistent velocity below the steady state velocity.

In an attempt to better investigate the interaction, the 2X system with the oxide added on was tested by applying the shear in increments at the same rate as used before, but with 10 ps of holding at certain strain values before the next increment. This simulation was performed at 300 K. Each increment was shown to result in the two dislocations moving apart from each other, only to stop at specific distances during the hold periods. The distance between the two dislocations at the end of each hold period is plotted vs. the applied strain in Figure 5.10(a) along with the predicted separation behavior given by Equation 5.6 showing a good agreement between the two. The separation distance is seen to initially be slightly greater than what is predicted, but then becomes slightly less than predicted at higher strains. While this is most likely just scatter, it could indicate a change in the interactions.

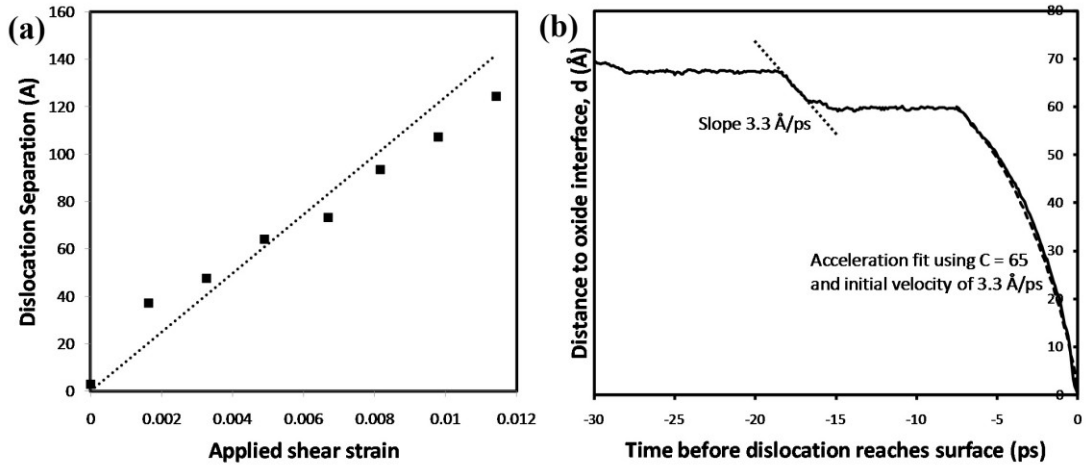

Figure 5.10: (a) Dislocation separation vs. applied shear for incremental loading showing behavior consistent with Equation 5.7. (b) The acceleration of a dislocation to the oxide interface during incremental loading. The movement of the dislocation both during the final strain increment and the previous increment are shown. A fit was found for the free surface attraction.

As the dislocations were moved by this incremental shearing, it was noticed that both did not move at the same rate and one approached the oxide interface sooner than the other. The dislocation that was closer to the free surface was observed to accelerate towards the interface immediately following the last incremental strain. The dislocation was 60 Å from the interface prior to the last strain increment, while the total predicted distance for both dislocations to move due to this increment was 20 Å. This indicates that the interface clearly had an attractive effect.

The dislocation's position with respect to the interface is plotted in Figure 5.10(b) showing not only the movement due to the last strain increment, but also the prior increment. Using the same acceleration model as was used with the free surface attraction, a good fit was found for the attracting by setting the initial velocity to 3.3 Å/ps and the attraction constant, C, to 65. The initial velocity was set to match what was observed with the previous increment. This was done as an alternative run that did not apply the final strain saw the dislocation remain at 60 Å away from the interface for over 100 ps. Thus, the motion due to the final strain increment was necessary for the attraction of the interface to occur. The value of the attraction constant is roughly 1/4-1/5 the constant found for the free surface indicating that oxide attraction is indeed weaker and consistent with the 1/3-1/4 strength predicted by the modulus based image force model.

Simulations were also performed in the zero applied strain conditions. With simulations both holding the shear strain at zero and simulations leaving the y surfaces unconstrained, the dislocations did not move. This was true for both the systems that featured oxides both farther away and at the same distance from the dislocations as the

dislocations were from the free surfaces in the free surface simulations. Within these simulations, the dislocations barely moved from their initial positions and were not seen to separate due to their repulsion of each other. Therefore, there is a large driving force within the system preventing the dislocations from moving. As the only difference between this simulation setup and the free surface simulation setup is the oxide at the surfaces, it stands to reason that the presence of the oxide has a long range repulsive effect on the dislocations for this system design.

Figure 5.11: An image of the 4X system with an oxide placed at the same distance as the free surfaces in the earlier simulations. This image was taken from an unconstrained run after 50 ps showing that the dislocations have not moved from their initial positions.

The most likely explanation for the long range repulsion is the result of the oxide interfaces placing the silicon surfaces in tension. The oxide is grown by inserting oxygen atoms into the silicon lattice. Because of this, the oxide wants to expand in all three directions, but is constrained by the silicon and the system design. While both x surfaces are in tension, the center of the silicon block is less affected resulting in a stress

gradient. This is noticeable in the unconstrained y surface simulations as a curvature is observed in the y surface. Stress gradients require that there be shear stresses within the simulation that can interact with the dislocations. For this simulation shape and design, the resulting shear stresses act as a long range repulsive force on the dislocations.

The interaction of the oxide layer on the dislocations therefore depends on the stress state that the oxide imposes on the system. As the image force attraction was found to be short ranged and weak, the longer range stress based interaction is therefore the most important effect of the oxide on the dislocations. The resulting stresses due to the oxide depend highly upon the system geometry and conditions. Therefore, the exact effect that the oxide has on the motion of the dislocations will be different for different systems.

As the long range interaction was found to be repulsive for the systems studied here, it follows that this can be the case for many systems, especially those that are similar. With a strong repulsive effect, the oxide will hinder the motion of dislocations towards the interfaces making it more likely for a large number of dislocations to be present within a volume of the material. In turn, this would allow for the measured hardness of the spheres to increase, as discussed in Chapter 4. If the stress state associated with the oxide on silicon spheres results in a strong enough repulsive effect, it could account for the experimentally observed hardness.

5.6 Summary

Dislocation interactions within diamond cubic silicon were investigated with molecular dynamics by using a variety of system configurations. It was found that with

a proper understanding of the effects that the defined simulation conditions have upon the measured response, the simulated behavior of the dislocations can be directly compared to the elasticity based theory. This comparison showed that at least qualitatively the simulated dislocation interactions match with the corresponding interactions from elementary dislocation theory. Because of this, these simulations or similarly designed simulations can provide an effective method for studying the nature of dislocations at the atomic scale.

Using periodic boundary conditions, two edge dislocations of opposite sign on parallel slip planes were shown to repulse each other when initially placed directly above and below each other. The motion of the dislocations in response to an externally applied shear strain was also shown to be consistent with geometrically necessary strain relaxation. A maximum limit for the dislocation velocity was also observed.

Introducing free surfaces into the system showed that the dislocations were attracted towards it. The free surface attraction was shown to be proportional to the inverse of the distance between the free surface and the position of the dislocation. This relationship matches with the image force model used in dislocation theory. The motion of the dislocation is shown to be impeded by the Peierls barrier when the system temperature is below 10 K.

Attraction of a dislocation to an oxide interface was also observed and was measured to be roughly 1/4-1/5 the strength of the attraction to a free surface. This corresponds to the image force model, which predicts that the attraction should be around 1/3-1/4 the free surface attraction. However, the oxide attraction appears to only

be a short range behavior as the dislocation had to be coerced into being less than 60 Å from the interface before the attraction occurred.

At larger distances, the presence of the oxide appears to have a repulsive effect on the dislocations. Constant strain rate simulations of the larger systems showed the dislocations moving at steady rates smaller than what was expected when far from the interface. In addition, the dislocations in the unconstrained systems did not move at all, not even to separate due to the dislocation-dislocation repulsion. This long range repulsion cannot be explained with the image force model and appears to result from the stress field due to the presence of the oxide. Understanding the effect of the oxide on the behavior of the dislocations within a Si/SiO_2 system therefore greatly depends on the stresses due to the oxide and the geometry of the system.

Chapter Six: Summary and Conclusions

6.1 Research Summary

The various molecular dynamics simulations that were reported here present a more comprehensive description of the yielding behaviors that occur within silicon nanoparticles. Chapter 3 used the Tersoff potential to investigate the possible transformation to the β-Sn phase that can occur within these spheres. The Stillinger-Weber potential was used in Chapter 4 and allowed for a study of how dislocations relate to the experimentally observed yielding and hardening behaviors. Chapter 5 continued the analysis of dislocations within silicon using a highly modified version of the Stillinger-Weber potential to measure the interactions active on the dislocations, including the effect of free surfaces and oxide interfaces.

Results with the Tersoff potential in Chapter 3 showed that the transformation from DC silicon to β-Sn silicon can occur within the spheres at high strains. However, this transformation is shown to be highly dependent on the crystallographic direction that the compression is applied along. For 10 nm spheres, no β-Sn was observed for [110] and [111] compressed spheres below 40% strain, whereas [100] compressed spheres showed evidence of β-Sn starting to form around 20% strain. Given the random orientation of experimentally compressed spheres, β-Sn will only form within favorably oriented spheres of comparable sizes.

Additionally, the high hardness values previously reported with the simulated compression of Tersoff silicon nanospheres was shown to not correspond with the β-Sn

transformation. Instead, the high hardness was due to the simulations running at or near 0 K as simulations at 300 K resulted in considerably lower values. The presence of β-Sn was shown to have an effect on the hardening behavior, but only at strains greater than 0.4 and for the favorably oriented [100] compressed sphere.

The Stillinger-Weber results in Chapter 4 showed that dislocations can explain both the hardening and yielding properties observed within the experimentally compressed spheres. Use of this potential resulted in the formation of a considerable number of dislocations. Compression along the [100] orientation was the most favorable for dislocations, and these dislocations were observed to nucleate on {110} planes. This {110} nucleation was an unexpected result and was found to occur due to the pathway for homogeneous nucleation on this plane being comparable to the pathway on the expected {111} planes with the Stillinger-Weber potential. More accurate quantum based calculations showed the {110} nucleation to be much less favorable and therefore their appearance is likely an artifact of using the Stillinger-Weber potential.

Even with this possibly unrealistic nucleation process, the vast number of dislocations observed with this potential allowed for the effect of their presence to be related to the observed experimental behaviors. Load drops were seen to correspond to dislocation motion indicating that the experimentally observed displacement excursions can be due to dislocation yielding. The dislocations that formed traveled through the spheres until they reached a free surface resulting in yielding deformation and a softening behavior. However, for a short period just after the initial yield point when the number of dislocations contained within the sphere is increasing, regions of hardening and consistent hardness were seen. The slope of this hardening behavior was

directly compared to the experimentally observed hardening showing a decent agreement. The experiments show hardening even at low strains (0.1-0.2) which can be explained by the dislocation hardening, but not by the β-Sn hardening.

Chapter 5 looked at the interactions that affect dislocations in silicon. The primary motivating factor for this work was the limitation of the spheres used in Chapters 3 and 4 in that they lacked an oxide layer. This lack of an oxide layer allowed for the vast number of dislocations that were observed in the Stillinger-Weber spheres of Chapter 4 being able to reach the surface and disappear. It was predicted that had there been an oxide layer on the surface of the spheres, the dislocations contained within would have either been confined within the sphere, or at least slightly more impeded. With more dislocations contained in the sphere, the hardening associated with those dislocations would be more prevalent and last for a longer period of time. As opposed to simply placing an oxide on the outside of one of these spheres, it was deemed more appropriate to investigate simpler more repeatable simulations first to determine the effectiveness of the oxide.

With a proper understanding of how the conditions imposed upon the system affect the nature of the dislocation interactions, simulations of this type were shown to be effective and comparable to the theoretical models. Tests without the oxides showed the presence of dislocation-dislocation repulsion, the correct nature of image force attractions to free surfaces, and the response due to an applied shear strain. Additionally, an upper limit was found for the dislocation velocities and simulations below 10 K showed effects of the Peierls barrier.

As for simulations with the oxide, the resulting response of the dislocations was more complex than what was predicted with the image force based theory. A weak attraction of a dislocation to one of the oxide interfaces was observed and seen to be consistent with the image force model. However, the long range behavior of the dislocations differed greatly and the presence of the oxide was repulsive to the dislocations. The repulsion appears to result from the stress state associated with the oxide creating a tensile stress within the surface of the silicon. It was therefore concluded that to obtain a proper understanding of how the presence of the oxide affects dislocations within a given silicon system, it is less important to know the image force attraction than it is the oxide induced stress state.

6.2 Conclusions

Altogether, the results presented in this thesis present an interesting description of the yielding associated with compressed silicon nanoparticles. There appears to be some size dependence to the specific observed yielding mechanisms. At the smallest sizes, there is expected to be a cutoff size for dislocation nucleation to occur. With the Stillinger-Weber potential, this cutoff was seen to be around the 5 nm diameter size as at most one dislocation was found in spheres of this size. The Tersoff potential showed a much larger cutoff diameter occurring between 10-20 nm.

At sizes slightly larger than the dislocation cutoff, the results here show that the behavior of the experimentally compressed nanospheres are best described by dislocations being the dominant yielding mechanism. The low strain hardening and the displacement excursions can both be explained by dislocation activity. β-Sn is not

expected to have a large presence in spheres within this size range as the Tersoff simulations show it to only form for favorable orientations.

While further increases in size were not studied here, the behavior can be inferred. β-Sn formation was highly dependent on the compression direction for the nanospheres, but has been observed to form during nanoindentation of bulk silicon [22-25]. This indicates that as the size of the particles increases, the β-Sn transformation becomes more likely.

6.3 Recommended Future Work

As comprehensive as the study of the silicon nanoparticles was here in that various sizes, potentials, temperatures and orientations were studied, there are still questions that could be answered with further simulations. Ideally, one would want to investigate larger spheres using both potentials. Larger Stillinger-Weber spheres would allow for even more dislocations to form and allow for the interactions between the dislocations to be better studied. Results from the Tersoff potential at larger sizes also should be intriguing as the 20 nm run at 300 K revealed an interesting change in the yielding behavior with dislocations nucleating and β-Sn forming outside of the particle's center. These large scale Tersoff spheres offer the opportunity of investigating dislocations and β-Sn simultaneously within these spheres.

It would also be incredibly useful for the larger spheres to be compressed along directions other than the [100] crystallographic direction. These alternate orientations could be beneficial as the {110} plane dislocations would be less favorable for the Stillinger-Weber potential better matching with what is observed experimentally. Work

with the Tersoff potential would also show whether dislocations or β-Sn would form for these alternate orientations at t larger sizes and larger displacements. Knowing this would go a long way to deciphering the experimental results.

The work in Chapter 5 with the dislocation interaction simulations was only an initial investigation of whether this or a similar technique could be used to better understand dislocations. As this was shown to be possible, it opens up the door for much continued work. Further studies should be done to find more quantitative descriptions of the interaction forces acting upon the dislocations. This would allow for different properties related to the material and the dislocations to be determined. A series of simulations could also conceivably be used to compare the interactions to the elasticity based dislocation theory models or to determine expressions for interactions in which determining the elasticity model is too difficult.

The dislocation simulations also serve as a basis for designing new systems for studying other dislocation related properties. For example, other material systems may be investigated, including ones where a known repulsive interface is present. Multiple dislocations can be placed on the same slip plane resulting in a dislocation pile-up. This pile-up is an important factor in grain boundary and second phase hardening mechanisms.

In addition to these direct continuations of the research presented here, two more complex research studies are proposed. The first is the inevitable combination of the spherical particle simulations with the oxide interaction simulations. The results of the particle compression simulations showed that the experimentally observed hardening could be explained by dislocation yielding. However considerable hardening and large

hardness values were not observed, which was attributed to the dislocations disappearing at the oxide free particle surface. The dislocation interaction simulations did show that the oxide can repulse dislocations due to the stress state caused by the presence of the oxide.

But in order to determine if the stress state associated with the oxide on the spheres is repulsive, further work is necessary. Continuum based calculations can be used to analyze the stresses introduced by the oxide in order to determine how they should affect dislocations. This can be done for systems comparable to both the dislocation interaction simulations and a spherical particle with an appropriate oxide.

In the end, it would be considerably useful in order to run at least one molecular dynamics simulation of a particle with an oxide layer created in a manner similar to what was used here. Any simulation would be considerably computationally expensive for a number of reasons. The Watanabe potential is more complex than the basic Stillinger-Weber potential making it slower to calculate. In addition, any simulation would have to be larger than 20 nm in order to see considerable dislocations to investigate the hardening. The particle size must also be adjusted as the growth of the oxide decreases the effective size of the silicon crystal. Finally, the oxide growing routine takes considerable time to grow a realistically thick oxide. This last step could be avoided by using a faster but possibly less realistic method for creating the oxide.

The final major proposal to future work involves looking at the ReaxFF potential [78-80] as an alternative potential for Si and Si-SiO_2 systems. The ReaxFF potential was initially designed to better match the quantum behavior of numerous atomic materials by being conditionally dependent on the number and composition of

the neighboring atoms. This conceivably results in a more realistic description of silicon and the oxide than can be obtained with the simpler potentials. The ReaxFF potential has also been shown to give excellent agreement between propagation of cracks in tension within silicon with quantum based models [78, 79].

The main downside with the ReaxFF potential is that it is considerably more computationally intense than the simpler empirical potentials. The crack propagation studies circumvented this by using the Tersoff potential for the silicon atoms in the bulk crystal and the ReaxFF potential only for the atoms at or around the crack and resulting free surface. This hybrid method requires that the boundary between the two potentials must be constantly updated as the crack propagates. Additional challenges arise for systems with compressive loads as the likelihood for plastic yielding becomes more pronounced. The hybrid method then becomes a liability as the boundary between the two potentials could interfere with the plastic behavior. Therefore, it would be preferable to use ReaxFF for the whole sphere.

Even with these considerations, the ReaxFF potential offers considerable promise for more realistic studies at the cost of being computationally intense. Initial work with this potential would involve comparing the elastic and plastic properties of the ReaxFF potential with other potentials and quantum based calculations. Once the elastic constants, the dislocation characteristics, and the phase transformation nature of the potential have been identified, larger scale simulations of the nanostructures can be accomplished.

References

[1] W. M. Miller, D. M. Tanner, S. L. Miller, K. A. Peterson, Sandia National Laboratories, **1998**, 7.

[2] http://www.sandia.gov/news-center/news-releases/2005/gen-science/mems.html **2005**.

[3] A. R. Beaber, L. J. Qi, J. Hafiz, P. H. McMurry, J. V. R. Heberlein, W. W. Gerberich, S. L. Girshick, *Surface and Coatings Technology* **2007**, *202*, 871.

[4] W. W. Gerberich, W. M. Mook, M. J. Cordill, C. B. Carter, C. R. Perrey, J. Heberlein, S. L. Girshick, *International Journal of Plasticity* **2005**, *21*, 2391.

[5] W. C. Oliver, G. M. Pharr, *Journal of Materials Research* **1992**, *7*, 1564.

[6] H. Hertz, *Journal fur die reine und angewandte Mathematik* **1882**, *92*, 156.

[7] W. W. Gerberich, W. M. Mook, C. R. Perrey, C. B. Carter, M. I. Baskes, R. Mukherjee, A. Gidwani, J. Heberlein, P. H. McMurry, S. L. Girshick, *Journal of the Mechanics and Physics of Solids* **2003**, *51*, 979.

[8] M. R. VanLandingham, T. F. Juliano, M. J. Hagon, *Measurement Science and Technology* **2005**, *16*, 2173.

[9] J. F. Cannon, *Journal of Physical and Chemical Reference Data* **1974**, *3*, 781.

[10] J. Crain, G. J. Ackland, J. R. Maclean, R. O. Piltz, P. D. Hatton, G. S. Pawley, *Physical Review B* **1994**, *50*, 10343.

[11] J. C. Jamieson, *Science* **1963**, *139*, 762.

[12] M. I. McMahon, R. J. Nelmes, *Physical Review B* **1993**, *47*, 8337.

[13] H. Olijnyk, S. K. Sikka, W. B. Holzapfel, *Physics Letters* **1984**, *103A*, 137.

[14] R. S. Wentorf, Jr., J. S. Kasper, *Science* **1963**, *25*, 338.

[15] R. O. Piltz, J. R. Maclean, S. J. Clark, G. J. Ackland, P. D. Hatton, J. Crain, *Physical Review B* **1995**, *52*, 4072.

[16] Y.-X. Zhou, F. Buehler, J. R. Sites, I. L. Spain, *Solid State Communications* **1986**, *59*, 679.

[17] L. L. Boyer, E. Kaxiras, J. L. Feldman, J. Q. Broughton, M. J. Mehl, *Physical Review Letters* **1991**, *67*, 715.

[18] F. H. Stillinger, T. A. Weber, *Physical Review B* **1985**, *31*, 5262.

[19] D. E. Kim, S. I. Oh, *Journal of Applied Physics* **2008**, *104*, 013502(6).

[20] J. Tersoff, *Physical Review B* **1988**, *37*, 6991.

[21] J. Tersoff, *Physical Review B* **1988**, *38*, 9902.

[22] D. E. Kim, S. I. Oh, *Nanotechnology* **2006**, *17*, 2259.

[23] Y.-H. Lin, T.-C. Chen, *Applied Physics A* **2008**, *92*, 571.

[24] Y.-H. Lin, S.-R. Jian, Y.-S. Lai, P.-F. Yang, *Nanoscale Research Letters* **2008**, *3*, 71.

[25] C. F. Sanz-Navarro, S. D. Kenny, R. Smith, *Nanotechnology* **2004**, *15*, 692.

[26] D. H. Alsem, C. L. Muhlstein, E. A. Stach, R. O. Ritchie, *Scripta Materiala* **2008**, *59*, 931.

[27] V. Domnich, Y. Gogotsi, S. Dub, *Applied Physics Letters* **2000**, *76*, 2214.

[28] J.-i. Jang, M. J. Lance, S. Wen, T. Y. Tsui, G. M. Pharr, *Acta Materialia* **2005**, *53*, 1759.

[29] T. Juliano, V. Domnich, Y. Gogotsi, *Journal of Materials Research* **2004**, *19*, 3099.

[30] A. Kailer, Y. G. Gogotsi, K. G. Nickel, *Journal of Applied Physics* **1997**, *81*, 3057.

[31] A. B. Mann, D. van Heerden, J. B. Pethica, T. P. Weihs, *Journal of Materials Research* **2000**, *15*, 1754.

[32] K. Wu, X. Q. Yan, M. W. Chen, *Applied Physics Letters* **2007**, *91*, 101903(3).

[33] P.-F. Yang, S.-R. Jian, Y.-S. Lai, T.-H. Chen, R.-S. Chen, "Nanoindentation-induced phase transformation of silicon", presented at *International Microsystems, Packaging, Assemby Conference*, Taiwan, **2006**.

[34] I. V. Gredneva, Y. V. Milman, V. I. Trefilov, *Physica Status Solidi a* **1972**, *14*, 177.

[35] D. R. Clarke, M. C. Krull, P. D. Kirchner, R. F. Cook, B. J. Hockey, *Physical Review Letters* **1988**, *60*, 2156.

[36] I. Zarudi, T. D. Nguyen, L. C. Zhang, *Applied Physics Letters* **2005**, *86*, 011922(3).

[37] I. Zarudi, L. C. Zhang, W. C. D. Cheong, T. X. Yu, *Acta Materialia* **2005**, *53*, 4795.

[38] I. Zarudi, L. C. Zhang, M. V. Swain, *Applied Physics Letters* **2003**, *82*, 1027.

[39] I. Zarudi, J. Zou, W. McBride, L. C. Zhang, *Applied Physics Letters* **2004**, *85*, 932.

[40] I. Zarudi, J. Zou, L. C. Zhang, *Applied Physics Letters* **2003**, *82*, 874.

[41] W. C. D. Cheong, L. C. Zhang, *Nanotechnology* **2000**, *11*, 173.

[42] Y.-H. Lin, T.-C. Chen, P.-F. Yang, S.-R. Jian, Y.-S. Lai, *Applied Surface Science* **2007**, *254*, 1415.

[43] G. S. Smith, E. B. Tadmore, E. Kaxiras, *Physical Review Letters* **2000**, *84*, 1260.

[44] J. Deneen, W. M. Mook, A. M. Minor, W. W. Gerberich, C. B. Carter, *Journal of Materials Science* **2006**, *41*, 4477.

[45] J. Deneen Nowak, W. M. Mook, A. M. Minor, W. W. Gerberich, C. B. Carter, *Philosophical Magazine* **2007**, *87*, 29.

[46] W. W. Gerberich, J. Michler, W. M. Mook, R. Ghisleni, F. Ostlund, D. D. Stauffer, R. Ballarini, *Journal of Materials Research* **2008**, *24*, 898.

[47] W. M. Mook, J. D. Nowak, C. R. Perrey, C. B. Carter, R. Mukherjee, S. L. Girshick, P. H. McMurry, W. W. Gerberich, *Physical Review B* **2007**, *75*, 214112.

[48] J. D. Nowak, A. R. Beaber, O. Ugurlu, W. W. Gerberich, O. L. Warren, *Microscopy and Microanalysis* **2007**, *15*, 722.

[49] J. D. Nowak, A. R. Beaber, O. Ugurlu, S. L. Girshick, W. W. Gerberich, *Scripta Materiala* **2010**, *62*, 819.

[50] A. M. Minor, E. T. Lilleodden, M. Jin, E. A. Stach, D. C. Chrzan, J. W. Morris Jr., *Philosophical Magazine* **2005**, *85*, 323.

[51] F. Ostlund, K. Rzepiejewska-Malyska, K. Leifer, L. M. Hale, Y. Tang, R. Ballarini, W. W. Gerberich, J. Michler, *Advanced Functional Materials* **2009**, *19*, 2439.

[52] D. Stauffer, W. W. Gerberich, *Unpublished* **2011**.

[53] J. F. Justo, R. D. Menezes, L. V. C. Assali, *Physical Review B* **2007**, *75*, 045303.

[54] P. W. Leu, A. Svizhenko, K. Cho, *Physical Review B* **2008**, *77*, 235305.

[55] R. D. Menezes, J. F. Justo, L. V. C. Assali, *Physica Status Solidi a* **2007**, *204*, 951.

[56] M. Menon, D. Srivastava, *Physical Review B* **2004**, *70*, 125313.

[57] Z. Yang, Z. Lu, Y.-P. Zhao, *Journal of Applied Physics* **2009**, *106*, 023537.

[58] K.-C. Fang, C.-I. Weng, S.-P. Ju, *Journal of Nanoparticle Research* **2009**, *11*, 581.

[59] P. Valentini, T. Dumitrica, *Physical Review B* **2007**, *75*, 224106.

[60] P. Valentini, W. W. Gerberich, T. Dumitrica, *Physical Review Letters* **2007**, *99*, 175701.

[61] N. Zhang, Q. Deng, Y. Hong, L. Xiong, S. Li, M. Strasberg, W. Yin, Y. Zou, C. R. Taylor, G. Sawyer, Y. Chen, *Journal of Applied Physics* **2011**, *109*, 063534.

[62] J. E. Jones, *Proceedings of the Royal Society of London. Series A* **1924**, *106*, 463.

[63] J. E. Lennard-Jones, *Transactions of the Faraday Society* **1929**, *25*, 668.

[64] W. C. Swope, H. C. Andersen, P. H. Berens, K. R. Wilson, *Journal of Chemical Physics* **1982**, *76*, 637.

[65] H. Balamane, T. Halicioglu, W. A. Tiller, *Physical Review B* **1992**, *46*, 2250.

[66] J. R. Ray, *Computer Physics Reports* **1988**, *8*, 109.

[67] I. P. Batra, F. F. Abraham, S. Ciraci, *Physical Review B* **1987**, *35*, 9552.

[68] R. A. Brown, D. Maroudas, T. Sinno, *Journal of Crystal Growth* **1994**, *137*, 12.

[69] T. Sinno, Z. K. Jiang, R. A. Brown, *Applied Physics Letters* **1996**, *68*, 3028.

[70] J. Godet, L. Pizzagalli, S. Brochard, P. Beauchamp, *Journal of Physics: Condensed Matter* **2003**, *15*, 6943.

[71] J. Tersoff, *Physical Review Letters* **1986**, *56*, 632.

[72] Z. Jiang, R. A. Brown, *Chemical Engineering Science* **1994**, *49*, 2991.

[73] Z. Jiang, R. A. Brown, *Physical Review Letters* **1995**, *74*, 2046.

[74] G. J. Kramer, N. P. Farragher, B. W. H. van Beest, R. A. van Santen, *Physical Review B* **1991**, *43*, 5068.

[75] B. W. H. van Beest, G. J. Kramer, R. A. van Santen, *Physical Review Letters* **1990**, *64*, 1955.

[76] T. Watanabe, H. Fujiwara, H. Noguchi, T. Hoshino, I. Ohdomari, *Japanese Journal of Applied Physics* **1999**, *38*, L366.

[77] T. Watanabe, I. Ohdomari, *Thin Solid Films* **1999**, *343-344*, 370.

[78] M. J. Buehler, H. Tang, A. C. T. van Duin, W. A. I. Goddard, *Physical Review Letters* **2007**, *99*, 165502.

[79] M. J. Buehler, A. C. T. van Duin, W. A. I. Goddard, *Physical Review Letters* **2006**, *96*, 095505.

[80] A. C. T. Van Duin, A. Strachan, S. Stewman, Q. Zhang, X. Xu, W. A. I. Goddard, *Journal of Physical Chemistry A* **2003**, *107*, 3803.

[81] P. Ganster, G. Treglia, A. Saul, *Physical Review B* **2010**, *81*, 045315.

[82] J. Dalla Torre, J.-L. Bocquet, Y. Limoge, J.-P. Crocombette, E. Adam, G. Martin, T. Baron, P. Rivallin, P. Mur, *Journal of Applied Physics* **2002**, *92*, 1084.

[83] LAMMPS Molecular Dynamics Simulator. http://lammps.sandia.gov/.

[84] W. G. Hoover, *Physical Review A* **1985**, *31*, 1695.

[85] L. M. Hale, X. Zhou, J. A. Zimmerman, N. R. Moody, R. Ballarini, W. W. Gerberich, *Computational Materials Science* **2011**, *In Press*.

[86] J. A. Zimmerman, C. L. Kelchner, P. A. Klein, J. C. Hamilton, S. M. Foiles, *Physical Review Letters* **2001**, *87*, 165507.

[87] K. L. Johnson, *Contact Mechanics*, Cambridge University Press, Cambridge, MA **1985**.

[88] D. D. Stauffer, L. M. Hale, A. R. Beaber, O. Ugurlu, J. D. Nowak, S. L. Girshick, R. Ballarini, W. W. Gerberich, *In preparation* **2011**.

[89] M. Vergeles, A. Maritan, J. Koplik, J. Banavar, *Physical Review E* **1997**, *56*, 2626.

[90] T. Frauenheim, F. Weich, T. Kohler, S. Uhlmann, D. Porezag, G. Seifert, *Physical Review B* **1995**, *52*, 11492.

[91] R. Rurali, E. Hernandez, *Computational Materials Science* **2003**, *28*, 85.

[92] Y.-M. Juan, E. Kaxiras, *Philosophical Magazine A* **1996**, *74*, 1367.

[93] J. Weertman, J. R. Weertman, *Elementary Dislocation Theory*, Oxford University Press, New York **1964**.

Appendix One: Identification Parameter Calculation

This appendix contains the code for the program that performed calculations to help in the analysis of the sphere compressions. The program opens and reads the atomic configuration files created by LAMMPS using the dump command with the atom style. With the atomic positions, it creates a nearest neighbor list for all of the atoms and calculates the slip vector (slp), the instantaneous slip vector (islp), the coordination number (coor), and the angular parameters (ang) that were used in the identification of defects. These parameters are then saved in separate files in a comparable format to the LAMMPS dump files. In addition, the log files for the LAMMPS runs were also open and read allowing for the contact area, strain and contact stress due to the compression to be calculated.

As most of the parameters investigated involved the creation of a list of nearest neighbors, the time to run this program can be high for large atomic systems. In order to reduce this run time, only a complete neighbor list is calculated from the initial step. All other subsequent neighbor lists are created by using the neighbor lists at the previous step as a basis. Each subsequent list only looked at atoms that were previously neighbors of a given atom, neighbors of the neighbors for that atom, and neighbors of the neighbors' neighbors. This greatly reduced the computation time and only 8 out of the 209121 atoms of the 20 nm spheres were observed to have values different from what was found by calculating the full neighbor list every time.

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <math.h>
#include <string>
#include <iomanip>
#include <sstream>
using namespace std;

#define PI 3.14159265
class Vect
{
public:
        Vect();
        Vect(double[3]);
        void set(double[3]);
        void set(double,double,double);
        double show(int);
        double coord[3], mag;
        Vect operator+ (Vect);
        Vect operator- (Vect);
        Vect operator* (double);
        Vect operator/ (double);
        Vect& operator=(const Vect&);
};
ostream &operator <<(ostream &out, Vect &vec)
{
        out << fixed << setprecision (5);
        out << vec.coord[0]*vec.mag << " " << vec.coord[1]*vec.mag << " " << vec.coord[2]*vec.mag;
        return out;
}

double dot(Vect param1, Vect param2)
{
    return param1.coord[0]*param1.mag*param2.coord[0]*param2.mag +
param1.coord[1]*param1.mag*param2.coord[1]*param2.mag +
param1.coord[2]*param1.mag*param2.coord[2]*param2.mag;
}

Vect Vect::operator+ (Vect param)
{
        double temp[3];
        Vect temp1;
        for (int i=0;i<3;i++)
                temp[i]=show(i)+param.show(i);
        temp1.set(temp);
        return temp1;
}
Vect Vect::operator- (Vect param)
{
        double temp[3];
        Vect temp1;
        for (int i=0;i<3;i++)
                temp[i]=show(i)-param.show(i);
        temp1.set(temp);
```

```
        return temp1;
}
Vect Vect::operator* (double param)
{
        Vect temp1 = *this;
        temp1.mag*=param;
        return temp1;
}
Vect Vect::operator/ (double param)
{
        Vect temp1 = *this;
        temp1.mag/=param;
        return temp1;
}
Vect& Vect::operator =(const Vect& param)
{
    for (int i=0;i<3;i++)
        coord[i]=param.coord[i];
        mag=param.mag;

        return *this;
}
Vect::Vect()
{
        coord[0] = 0;
        coord[1] = 0;
        coord[2] = 0;
        mag = 0;
}

Vect::Vect(double x[3])
{
        set(x);
}
void Vect::set(double at,double bt,double ct)
{
        double x[3];
        x[0]=at;
        x[1]=bt;
        x[2]=ct;
        set(x);
}

void Vect::set(double x[3])
{
        double temp;
        temp = x[0]*x[0]+x[1]*x[1]+x[2]*x[2];
        mag = sqrt(temp);
        if (mag==0)
        {
                for (int i=0;i<3;i++)
                        coord[i]=0;
        }
        else
        {
```

```
                for (int i=0;i<3;i++)
                        coord[i]=x[i]/mag;
        }
}
double Vect::show(int position)
{
        if (position>=0&&position<=2)
                return coord[position]*mag;
        else
                return 0;
}


class Atom
{
public:
        Vect pos;
        int type,id,neigh[12],nneighs;
        bool contact;
        Atom();
        void set(int, int, double [3]);
        void check(Atom &, double);
        void clear();
        Vect operator+ (Atom);
        Vect operator- (Atom);
        Atom& operator=(const Atom&);
};
ostream &operator <<(ostream &out, Atom &atomo)
{
        out << atomo.nneighs;
        return out;
}

Vect Atom::operator+ (Atom param)
{
        return pos + param.pos;
}
Vect Atom::operator- (Atom param)
{
        return pos - param.pos;
}
Atom::Atom()
{
        clear();
}

void Atom::clear()
{
        pos.set(0,0,0);
        id = 0;
        type = 0;
        nneighs=0;
        contact = false;
        for (int i=0;i<12;i++)
                neigh[i]=0;
```

```
}
void Atom::set(int name, int types, double x[3])
{
        pos.set(x);
        type = types;
        id = name;
}

void Atom::check(Atom &a, double cutoff)
{
        Vect temp;
        bool newcheck;
        temp = pos - a.pos;
        newcheck = true;
        if (temp.mag<=cutoff && id!=a.id)
        {
                for (int i=0;i<nneighs;i++)
                {
                    if (neigh[i]==a.id) newcheck=false;
                }
                if (newcheck)
                {
                    neigh[nneighs]=a.id;
                    nneighs++;
                    a.neigh[a.nneighs]=id;
                    a.nneighs++;
                }
        }
}
Atom& Atom::operator =(const Atom& param)
{
        pos=param.pos;
        type=param.type;
        id=param.id;
        nneighs=param.nneighs;
        for (int i=0;i<nneighs;i++)
            neigh[i]=param.neigh[i];
    for (int i=nneighs;i<8;i++)
            neigh[i]=0;
        contact=param.contact;

 return *this;
}

class Data
{
public:
        Data();
    void set(double[6]);
    double p[2], f[2], pe, r[2], s[2], d, e;
        void stress();
        void strain(double, double);
        Data& operator=(const Data&);
};
```

```
ostream &operator <<(ostream &out, Data &dat)
{
	out << setw(13) << dat.d << setw(13) << dat.e;
	out << setw(13) << dat.f[0] << setw(13) << dat.f[1];
	out << setw(13) << dat.s[0] << setw(13) << dat.s[1];
	out << setw(13) << dat.r[0] << setw(13) << dat.r[1];
	out << setw(13) << dat.pe;
	return out;
}

istream &operator >>(istream &in, Data &dat)
{
	double temp;
	in >> dat.p[0] >> dat.p[1] >> dat.f[0] >> dat.f[1] >> dat.pe >> temp;
	return in;
}

Data::Data()
{
	pe=0;
	d=0;
	e=0;
	for (int i=0;i<2;i++)
	{
		p[i]=0;
		f[i]=0;
		r[i]=0;
		s[i]=0;
	}
}

void Data::set(double data[5])
{
    p[0]=data[0];
    p[1]=data[1];
    f[0]=data[2];
    f[1]=data[3];
    pe=data[4];
}

void Data::stress()
{
	f[0]=f[0]*1.602177/1000;
	f[1]=f[1]*-1.602177/1000;
	for (int i=0;i<2;i++)
	{
		if (r[i]==0)
			s[i]=0;
		else
			s[i]=f[i]*100000/(PI*r[i]*r[i]);
	}
}

void Data::strain(double max, double min)
{
```

```
	d = (max-min-p[0]+p[1])*.5;
	if (d<0)
		d=0;
	e = 2*d/(max-min);
}

Data& Data::operator =(const Data& param)
{
	pe=param.pe;
	d=param.d;
	e=param.e;
	for (int i=0;i<2;i++)
	{
		p[i]=param.p[i];
		f[i]=param.f[i];
		r[i]=param.r[i];
		s[i]=param.s[i];
	}
 return *this;
}

//*************************************************************************************
*************************************************************************************
****
int main()
{
	int max, natoms, num, type, ccountt, ccountb, count, c, ncheck, nlog, delta, check, avcount,
angcount,dsteps;
	double x[3], hold1, hold2, hold3, cutoff1, cutoff2, amax, amin, tempdata[5], avdata[5],
temperature, angular, cosang;
	string name, filen, filel, files, filei, filec, filea, temp1, temp2, temp3, temp4, temp5, temp6,
temp7, lhold;
	Vect slip, islip, centert, centerb, bond1, bond2;
	stringstream file1, file2, file3, file4, file5, file6, logf, line;
	char nums[10], buf[1024];
	bool nlist;

	cutoff1 = 3;
	cutoff2 = 0.1;
	max = 23040000;
	delta = 40000;
	nlog = 5;
	dsteps = max/delta+2;
	nlist = true;
	amax = 0;
	amin = 0;

	name = "atom.";
	file1 << name << 0;
	filen = file1.str();
	file1.seekp(ios_base::beg);
	ifstream fin0(filen.c_str());

	fin0 >> temp1 >> temp2;
	fin0 >> temp1;
```

```
fin0 >> temp1 >> temp2 >> temp3 >> temp4;
fin0 >> natoms;
fin0 >> temp1 >> temp2 >> temp3;
fin0 >> temp1 >> temp2;
fin0 >> temp1 >> temp2;
fin0 >> temp1 >> temp2;
fin0 >> temp1 >> temp2 >> temp3 >> temp4 >> temp5 >> temp6 >> temp7;

Atom * atoms0 = new Atom[natoms]; //need for calculating slip vector
Atom * atomsp = new Atom[natoms]; //need for calculating instantaneous slip vector
cout << natoms << " ";
cout.flush();
//Read initial data
for (int i=0;i<natoms;i++)
{
	fin0 >> num >> type >> x[0] >> x[1] >> x[2];
	if (x[1]>amax)
		amax = x[1];
	if (x[1]<amin || amin==0)
		amin = x[1];
	atoms0[num-1].set(num,type,x);
}
fin0.close();
cout << "Data in" << endl << "Nearest neighbor progress:";
cout.flush();
if (nlist==false)
{
	//Compile nearest neighbor list
	for (int i=0;i<natoms;i++)
	{
		for (int j=0;j<i;j++)
			atoms0[i].check(atoms0[j],cutoff1);
		if (i%(natoms/10)==0)
		{
			cout << "=";
			cout.flush();
		}
	}
	cout << endl;

	ofstream nout("nlist.txt");
	for (int i=0;i<natoms;i++)
	{
		atomsp[i]=atoms0[i];
		nout << atoms0[i].id << " " << atoms0[i].nneighs << " ";
		for (int j=0;j<atoms0[i].nneighs;j++)
			nout << atoms0[i].neigh[j] << " ";
		nout << endl;
	}
	nout.close();
}
else
{
	//Read in nearest neighbor list
	ifstream nin("nlist.txt");
```

```
		for (int i=0;i<natoms;i++)
		{
			nin >> num;
			nin >> atoms0[num-1].nneighs;
			for (int j=0;j<atoms0[num-1].nneighs;j++)
				nin >> atoms0[num-1].neigh[j];
			atomsp[num-1]=atoms0[num-1];
			if (i%(natoms/10)==0)
			{
				cout << "=";
				cout.flush();
			}
		}
		cout << endl;

}
nums[0]='0';
nums[1]='1';
nums[2]='2';
nums[3]='3';
nums[4]='4';
nums[5]='5';
nums[6]='6';
nums[7]='7';
nums[8]='8';
nums[9]='9';
ncheck = 1;

Data * data = new Data[dsteps];
ofstream logout("log.txt");

avcount=0;
for (int j=1;j<=nlog;j++) //read from log.lammps files
{
		for (int n=0;n<5;n++)
		 avdata[n]=0;

	logf << "log" << j << ".lammps";
		filel = logf.str();
		logf.seekp(ios_base::beg);
		ifstream login(filel.c_str());
		cout << filel << endl;
		while(1)
		{
			login.getline(&buf[0],1024);
			if (login.eof()==true)
				break;
			for (int k=0;k<10;k++)
			{
				for (int p=0;p<10;p++)
				{
					if (buf[7]==nums[k] && buf[20]==nums[p])
					{
						line.str(string(buf));
						lhold=line.str();
```

```
                                                logout << lhold << endl;
                                                line.seekp(ios_base::beg);
                                                line >> count >> c >> tempdata[0] >> tempdata[1]
>> tempdata[2] >> tempdata[3] >> tempdata[4] >> temperature;
                                                line.seekg(ios_base::beg);

                                                if (c>avcount)
                                                {
                                                      check = (100*(count-(c-1)*delta))/delta;

                                                          if (check > 80)
                                                          {
                                                               for (int n=0;n<5;n++)
                                                                    avdata[n]+=tempdata[n];
                                                      }

                                                          if (check == 100)
                                                          {
                                                               for (int n=0;n<5;n++)
                                                                    avdata[n]=avdata[n]/20;

                                                               data[c-1].set(avdata);
                                                               data[c-1].strain(amax,amin);
                                                               for (int n=0;n<5;n++)
                                                                        avdata[n]=0;
                                                               avcount++;
                                                          }
                                                }

                                          }

                                    }
                              }
                        }
                        login.close();
                }
                logout.close();

                ofstream lout("data.txt");
                lout << "   c      disp(A)       strain    force1(uN)   force2(uN)";
                lout << " stress1(GPa) stress2(GPa)       a1(A)       a2(A)       pe(eV)" << endl;

                int * clistt = new int[30000];
                int * clistb = new int[30000];
                Atom * atoms1 = new Atom[natoms];

                for (int g=0;g<=max;g+=delta)       //For all timesteps
                {
                   if (g!=50)
                        {
                        for (int i=0;i<natoms;i++)
                                atoms1[i].clear();

                        c = g/delta;
```

```
file2 << name <<g;
filen = file2.str();
file2.seekp(ios_base::beg);
file3 << "slp." << g;
files = file3.str();
file3.seekp(ios_base::beg);
file4 << "islp." <<g;
filei = file4.str();
file4.seekp(ios_base::beg);
file5 << "coor." << g;
filec = file5.str();
file5.seekp(ios_base::beg);
file6 << "ang." << g;
filea = file6.str();
file6.seekp(ios_base::beg);

cout << filen << " " << files << " " << filei << " " << filec << " " << filea << endl;

ifstream fin(filen.c_str());
ofstream sout(files.c_str());
ofstream isout(filei.c_str());
ofstream coorout(filec.c_str());
ofstream angout(filea.c_str());

//copy through headers
for (int y=1;y<=8;y++)
{
        fin.getline(&(buf[0]),1024);
        temp1 = string(buf);
        sout << temp1 << endl;
        isout << temp1 << endl;
        coorout << temp1 << endl;
        angout << temp1 << endl;
}
fin.getline(&(buf[0]),1024);
sout << "ITEM: ATOMS id xslip yslip zslip slipmag" << endl;
isout << "ITEM: ATOMS id xislp yislp zislp islpmag" << endl;
coorout << "ITEM: ATOMS id coord" << endl;
angout << "ITEM: ATOMS id ang" << endl;

//set up for reading timestep data

ccountt = 0;
ccountb = 0;
centert.set(0,0,0);
centerb.set(0,0,0);

//Read timestep data
for (int i=0;i<natoms;i++)
{
        fin >> num >> type >> x[0] >> x[1] >> x[2];

        atoms1[num-1].set(num,type,x);
        if (c>0)
        {
```

```
                                        if ((data[c-1].p[0]-x[1])<cutoff2)
                                        {
                                                centert = centert+atoms1[num-1].pos;
                                                clistt[ccountt]=num-1;
                                                ccountt++;
                                        }
                                        if ((x[1]-data[c-1].p[1])<cutoff2)
                                        {
                                                centerb = centerb+atoms1[num-1].pos;
                                                clistb[ccountb]=num-1;
                                                ccountb++;
                                        }
                                }
                        }
                        cout << ccountt << " " << ccountb << endl;
                        fin.close();

        for (int i=0;i<natoms;i++)
                        {
                          for (int j=0;j<atomsp[i].nneighs;j++)                         //for all nearest
neighbors
                                {
                                        atoms1[i].check(atoms1[atomsp[i].neigh[j]-1],cutoff1);
                                        for (int k=0;k<atomsp[atomsp[i].neigh[j]-1].nneighs;k++)
//for all nearest neighbors of a nearest neighbor
                                        {
                                             atoms1[i].check(atoms1[atomsp[atomsp[i].neigh[j]-1].neigh[k]-
1],cutoff1);
                                             for (int l=0;l<atomsp[atomsp[atomsp[i].neigh[j]-1].neigh[k]-
1].nneighs;l++) //for all nneighbors of a nneighbor of a nneighbor.
                                             {
                                                  atoms1[i].check(atoms1[atomsp[atomsp[atomsp[i].neigh[j]-
1].neigh[k]-1].neigh[l]-1],cutoff1);
                                                }
                                        }
                                }
                        }

                        //Slip vector
                        for (int i=0;i<natoms;i++)
                        {
                          coorout << i+1 << " " << atoms1[i].nneighs << endl;
                          slip.set(0,0,0);
                                islip.set(0,0,0);

                                for (int k=0;k<atoms0[i].nneighs;k++)
                                        slip = slip + (atoms1[atoms0[i].neigh[k]-1]-atoms1[i])-
(atoms0[atoms0[i].neigh[k]-1]-atoms0[i]);
                                sout << i+1 << " " << slip << " " << slip.mag << endl;

                                for (int k=0;k<atomsp[i].nneighs;k++)
                                      islip = islip + (atoms1[atomsp[i].neigh[k]-1]-atoms1[i])-
(atomsp[atomsp[i].neigh[k]-1]-atomsp[i]);
                                isout << i+1 << " " << islip << " " << islip.mag << endl;
                        }
```

```
sout.close();
isout.close();
coorout.close();

//angle calculation and reset atomsp
for (int i=0;i<natoms;i++)
{
      atomsp[i]=atoms1[i];
          angular = 0;
          angcount = 0;
          for (int k=0;k<atoms1[i].nneighs;k++)
          {
                bond1 = atoms1[atoms1[i].neigh[k]-1]-atoms1[i];
                for (int j=0;j<k;j++)
                      {
                            bond2 = atoms1[atoms1[i].neigh[j]-1]-atoms1[i];
                                cosang = dot(bond1,bond2)/(bond1.mag*bond2.mag);
                                angular += (cosang+(1.0/3.0))*(cosang+(1.0/3.0));
                                angcount++;
                      }
          }
          if (angcount==0)
            angular = 0;
          else
            angular = angular/angcount;
          angout << i+1 << " " << angular << endl;
}
angout.close();

//contact diameter
hold1 = 0;
hold2 = 0;
hold3 = 0;
if (ccountt > 0)
{
          centert=centert/ccountt;
          hold3 = 0;
          for (int p=0;p<ccountt;p++)
          {
                    hold1 = atoms1[clistt[p]].pos.show(0)-centert.show(0);
                    hold2 = atoms1[clistt[p]].pos.show(2)-centert.show(2);
                    hold3 += hold1*hold1+hold2*hold2;
          }
          data[c-1].r[0] = sqrt(2.0*hold3/ccountt);
}
if (ccountb > 0)
{
          centerb=centerb/ccountb;
          hold3 = 0;
          for (int p=0;p<ccountb;p++)
          {
                    hold1 = atoms1[clistb[p]].pos.show(0)-centerb.show(0);
                    hold2 = atoms1[clistb[p]].pos.show(2)-centerb.show(2);
                    hold3 += hold1*hold1+hold2*hold2;
          }
```

```
                        data[c-1].r[1] = sqrt(2.0*hold3/ccountb);
                }
                if (c>0)
                {
                        data[c-1].stress();
                        lout << setw(4) << c << data[c-1] << endl;
                }
                }
        }
        lout.close();
        delete [] atoms1;
        delete [] clistt;
        delete [] clistb;
        delete [] atomsp;
        delete [] atoms0;
        delete [] data;
        return 0;
}
```

Appendix Two Oxide Potentials

This appendix contains the code developed for the implementation of the Jiang and Brown, and the Watanabe et al. Si-SiO_2 atomistic potentials. The Jiang and Brown potential is named as the swbks potential for being a combination of the Stillinger-Weber and the BKS potentials, while the Watanabe potential is the named the sw_wnfho potential. To use these potentials, the .h and .cpp files need to be placed in the src folder of the LAMMPS program and LAMMPS needs to be rebuilt.

The Jiang and Brown potential is then ran in LAMMPS using the lines

```
pair_style          swbks 10.0
pair_coeff          * * /folder-location/SiO.swbks Si O
```

The Watanabe potential can be ran using

```
pair_style      sw/wfnho
pair_coeff          * * /folder-location/SiO.sw_wfnho Si(a) O
```

As can be seen, the coefficients of the parameters for both potentials are contained in separate folders. These parameter files are also included here.

"pair_sw_bks.h"

```
/* ----------------------------------------------------------------------
   LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
   http://lammps.sandia.gov, Sandia National Laboratories
   Steve Plimpton, sjplimp@sandia.gov

   Copyright (2003) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level LAMMPS directory.
------------------------------------------------------------------------- */

#ifdef PAIR_CLASS

PairStyle(swbks,PairSWBKS)

#else

#ifndef LMP_PAIR_SW_BKS_H
#define LMP_PAIR_SW_BKS_H

#include "pair.h"

namespace LAMMPS_NS {

class PairSWBKS : public Pair {
public:
        PairSWBKS(class LAMMPS *);
        PairSWBKS(class LAMMPS *);
        ~PairSWBKS();
        void compute(int, int);
        void settings(int, char **);
        void coeff(int, char **);
        double init_one(int, int);
        void init_style();

private:
        struct Param {
                double epsilon,sigma;
                double littlea,lambda,gamma,costheta;
                double biga,bigb;
                double powerp,powerq;
                double tol;
                double cut,cutsq;
                double sigma_gamma,lambda_epsilon,lambda_epsilon2;
                double c1,c2,c3,c4,c5,c6;
                int ielement,jelement,kelement;
        };
        double cut_lj_global;
        double **cut_lj,**cut_ljsq;
```

```
        double cut_coul,cut_coulsq;
        double **a,**rho,**c;
        double **rhoinv,**buck1,**buck2,**offset;
        double g_ewald;
        double cutmax;             // max cutoff for all elements
        int nelements;            // # of unique elements
        char **elements;           // names of unique elements
        int **swcheck;             // identifies which element is modeled with SW
        int *map,*qset;            // mapping from atom types to elements and charge param check
        Param swparams;              // parameter set for SW potential

        void allocate();
        void read_file(char *);
        void setup();
        double gij(double, double);
        void twobody(Param *, double, double &, int, double &);
        void threebody(Param *, double, double, double, double, double, double *, double *,double *,
double *, int, double &);
};

}

#endif
#endif
```

"pair_sw_bks.cpp"

```
/* ----------------------------------------------------------------------
   LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
   http://lammps.sandia.gov, Sandia National Laboratories
   Steve Plimpton, sjplimp@sandia.gov

   Copyright (2003) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level LAMMPS directory.
------------------------------------------------------------------------- */

/* ----------------------------------------------------------------------
   Spliced by Lucas Hale from pair_sw contributed by author: Aidan Thompson (SNL)
   and pair_buck_coul_long
------------------------------------------------------------------------- */

#include "math.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "pair_sw_bks.h"
#include "atom.h"
#include "kspace.h"
#include "neighbor.h"
#include "neigh_request.h"
#include "force.h"
#include "comm.h"
#include "memory.h"
#include "neighbor.h"
#include "neigh_list.h"
#include "memory.h"
#include "error.h"

using namespace LAMMPS_NS;

#define MAXLINE 1024
#define DELTA 4

#define MIN(a,b) ((a) < (b) ? (a) : (b))
#define MAX(a,b) ((a) > (b) ? (a) : (b))

#define EWALD_F   1.12837917
#define EWALD_P   0.3275911
#define A1        0.254829592
#define A2       -0.284496736
#define A3        1.421413741
#define A4       -1.453152027
#define A5        1.061405429
#define PIval        3.14159265359
```

```
/* ---------------------------------------------------------------------- */

PairSWBKS::PairSWBKS(LAMMPS *lmp) : Pair(lmp)
{
  single_enable = 0;
  one_coeff = 1;

  nelements = 0;
  elements = NULL;
  params = NULL;
}

/* ----------------------------------------------------------------------
   check if allocated, since class can be destructed when incomplete
---------------------------------------------------------------------- */

PairSWBKS::~PairSWBKS()
{
  if (elements)
    for (int i = 0; i < nelements; i++) delete [] elements[i];
  delete [] elements;
 // memory->sfree(swparams);

  if (allocated) {
    memory->destroy_2d_int_array(setflag);
            memory->destroy_2d_int_array(swcheck);
    memory->destroy_2d_double_array(cutsq);

    memory->destroy_2d_double_array(cut_lj);
    memory->destroy_2d_double_array(cut_ljsq);
    memory->destroy_2d_double_array(a);
    memory->destroy_2d_double_array(rho);
    memory->destroy_2d_double_array(c);
    memory->destroy_2d_double_array(rhoinv);
    memory->destroy_2d_double_array(buck1);
    memory->destroy_2d_double_array(buck2);
    memory->destroy_2d_double_array(offset);
    delete [] map;
            delete [] qset;
  }
}

/* ---------------------------------------------------------------------- */

void PairSWBKS::compute(int eflag, int vflag)
{
            int i,j,k,ii,jj,kk,inum,jnum,jnumm1,itag,jtag;
            int itype,jtype,ktype,swparam,posneg;
            double xtmp,ytmp,ztmp,delx,dely,delz,evdwl,ecoul,fpair,gsoft;
            double rsq,rsq1,rsq2,r,r2inv,r6inv,forcecoul,forcebuck,factor_coul,factor_lj;
            double delr1[3],delr2[3],fj[3],fk[3],grij,expm2,prefactor,t,erfc,rexp;
            int *ilist,*jlist,*numneigh,**firstneigh;

            evdwl = ecoul = 0.0;
```

```
if (eflag || vflag) ev_setup(eflag,vflag);
else evflag = vflag_fdotr = 0;

double **x = atom->x;
double **f = atom->f;
int *tag = atom->tag;
int *type = atom->type;
int nlocal = atom->nlocal;
int nall = nlocal + atom->nghost;
double *special_coul = force->special_coul;
double *special_lj = force->special_lj;
int newton_pair = force->newton_pair;
double qqrd2e = force->qqrd2e;

inum = list->inum;
ilist = list->ilist;
numneigh = list->numneigh;
firstneigh = list->firstneigh;

// Calculate the effective charge on the atoms

for (ii=0; ii< inum; ii++) {
        i = ilsit[ii];
        itype = map[type[i]];
        xtmp = x[i][0];
        ytmp = x[i][1];
        ztmp = x[i][2];
        jlist = firstneigh[i];
        jnum = numneigh[i];
        qe[i] = 0.0;

        if (qnaught[itype] > 0) posneg = 1;
        else if (qnaught[itype] < 0) posneg = -1;
        else posneg = 0;

        for (jj = 0; jj < jnum; jj++) {
                j = jlist[jj];
                jtype = type[j];

                if (itype!=jtype) {
                        delx = xtmp - x[j][0];
                        dely = ytmp - x[j][1];
                        delz = ztmp - x[j][2];
                        rsq = delx*delx + dely*dely + delz*delz;
                        r = sqrt(rsq);

                        if (r <= ro) qe[i] += posneg*qo;
                        else if (r <= rs) qe[i] += posneg*qo*(1+cos(PIval*(r-ro)/(rs-ro)))/2;
                }//end if different
        }//end for j
}//end for i

// calculate the energies and forces

for (ii = 0; ii < inum; ii++) {
```

```
i = ilist[ii];
itag = tag[i];
qtmp = qe[i];
itype = map[type[i]];
jlist = firstneigh[i];
jnum = numneigh[i];
xtmp = x[i][0];
ytmp = x[i][1];
ztmp = x[i][2];

qdel = posneg*(qe[i] - qnaught[itype]);
if (qdel > 0)
        eion = eo*exp(-1/qdel);
//ADD eion ENERGY !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

for (jj = 0; jj < jnum; jj++) {  //pair potentials
        j = jlist[jj];
        jtag = tag[j];
        jtype = map[type[j]];

        if (swcheck[itype,jtype])  // if atom pair is Stillinger-Weber type bonding
        {
                if (itag > jtag) {
                        if ((itag+jtag) % 2 == 0) continue;
                } else if (itag < jtag) {
                        if ((itag+jtag) % 2 == 1) continue;
                } else {
                        if (x[j][2] < ztmp) continue;
                        if (x[j][2] == ztmp && x[j][1] < ytmp) continue;
                        if (x[j][2] == ztmp && x[j][1] == ytmp && x[j][0] < xtmp)
continue;
                }

                delx = xtmp - x[j][0];
                dely = ytmp - x[j][1];
                delz = ztmp - x[j][2];
                rsq = delx*delx + dely*dely + delz*delz;

                if (rsq > swparams.cutsq) continue;

                twobody(&swparams,rsq,fpair,eflag,evdwl);
                fpair = fpair*gij(qe[i],qe[j]);
                evdwl = evdwl*gij(qe[i],qe[j]);

                f[i][0] += delx*fpair;
                f[i][1] += dely*fpair;
                f[i][2] += delz*fpair;
                f[j][0] -= delx*fpair;
                f[j][1] -= dely*fpair;
                f[j][2] -= delz*fpair;
                if (evflag)
ev_tally(i,j,nlocal,newton_pair,evdwl,0.0,fpair,delx,dely,delz);

        } else {                        // if not Stillinger-Weber, must be BKS
(buck_coul_long)
```

```
                        if (j < nall) factor_coul = factor_lj = 1.0;
                        else {
                                factor_coul = special_coul[j/nall];
                                factor_lj = special_lj[j/nall];
                                j %= nall;
                        }

                        delx = xtmp - x[j][0];
                        dely = ytmp - x[j][1];
                        delz = ztmp - x[j][2];
                        rsq = delx*delx + dely*dely + delz*delz;

                        if (rsq < cutsq[itype][jtype]) {
                                r2inv = 1.0/rsq;

                                if (rsq < cut_coulsq) {
                                        r = sqrt(rsq);
                                        grij = g_ewald * r;
                                        expm2 = exp(-grij*grij);
                                        t = 1.0 / (1.0 + EWALD_P*grij);
                                        erfc = t * (A1+t*(A2+t*(A3+t*(A4+t*A5)))) *
expm2;

                                        prefactor = qqrd2e * qtmp*qe[j]/r;
                                        forcecoul = prefactor * (erfc +
EWALD_F*grij*expm2);

                                        if (factor_coul < 1.0) forcecoul -= (1.0-
factor_coul)*prefactor;
                                } else forcecoul = 0.0;

                                if (rsq < cut_ljsq[itype][jtype]) {
                                        r6inv = r2inv*r2inv*r2inv;
                                        r = sqrt(rsq);
                                        rexp = exp(-r*rhoinv[itype][jtype]);
                                        forcebuck = buck1[itype][jtype]*r*rexp -
buck2[itype][jtype]*r6inv;
                                } else forcebuck = 0.0;

                                fpair = (forcecoul + factor_lj*forcebuck) * r2inv;

                                f[i][0] += delx*fpair;
                                f[i][1] += dely*fpair;
                                f[i][2] += delz*fpair;
                                if (newton_pair || j < nlocal) {
                                        f[j][0] -= delx*fpair;
                                        f[j][1] -= dely*fpair;
                                        f[j][2] -= delz*fpair;
                                }

                                if (eflag) {
                                        if (rsq < cut_coulsq) {
                                                ecoul = prefactor*erfc;
                                                if (factor_coul < 1.0) ecoul -= (1.0-
factor_coul)*prefactor;
                                        } else ecoul = 0.0;
                                        if (rsq < cut_ljsq[itype][jtype]) {
```

```
                                                    evdwl = a[itype][jtype]*rexp -
c[itype][jtype]*r6inv - offset[itype][jtype];
                                                    evdwl *= factor_lj;
                                                } else evdwl = 0.0;
                                            }

                                            if (evflag)
ev_tally(i,j,nlocal,newton_pair,evdwl,ecoul,fpair,delx,dely,delz);
                                        }//end if
                                    }//end swcheck=false
                                }//end for j

                                jnumm1 = jnum - 1;
                                for (jj = 0; jj < jnumm1; jj++) {  //Start 3-body terms
                                        j = jlist[jj];
                                        jtype = map[type[j]];
                                        if (!swcheck[itype,jtype]) continue;

                                        delr1[0] = x[j][0] - xtmp;
                                        delr1[1] = x[j][1] - ytmp;
                                        delr1[2] = x[j][2] - ztmp;
                                        rsq1 = delr1[0]*delr1[0] + delr1[1]*delr1[1] + delr1[2]*delr1[2];
                                        if (rsq1 > swparams.cutsq) continue;

                                        for (kk = jj+1; kk < jnum; kk++) {
                                                k = jlist[kk];
                                                ktype = map[type[k]];
                                                if (!swcheck[itype,ktype]) continue;

                                                delr2[0] = x[k][0] - xtmp;
                                                delr2[1] = x[k][1] - ytmp;
                                                delr2[2] = x[k][2] - ztmp;
                                                rsq2 = delr2[0]*delr2[0] + delr2[1]*delr2[1] + delr2[2]*delr2[2];
                                                if (rsq2 > params.cutsq) continue;

                                                threebody(&params,rsq1,rsq2,delr1,delr2,fj,fk,eflag,evdwl);

                                                gsoft = gij(qe[i],qe[j])*gij(qe[i],qe[k]);
                                                for (int y=0;y<3;y++) {
                                                        fj[y] = fj[y] * gsoft;
                                                        fk[y] = fk[y] * gsoft;
                                                }
                                                evdwl = evdwl * gsoft;

                                                f[i][0] -= fj[0] + fk[0];
                                                f[i][1] -= fj[1] + fk[1];
                                                f[i][2] -= fj[2] + fk[2];
                                                f[j][0] += fj[0];
                                                f[j][1] += fj[1];
                                                f[j][2] += fj[2];
                                                f[k][0] += fk[0];
                                                f[k][1] += fk[1];
                                                f[k][2] += fk[2];

                                                if (evflag) ev_tally3(i,j,k,evdwl,0.0,fj,fk,delr1,delr2);
```

```
                        }//end for k
                }//end for j
        }//end for i

        if (vflag_fdotr) virial_compute();
}

/* ----------------------------------------------------------------------
  Bond softening function
------------------------------------------------------------------------ */

double PairSWBKS::gij(double qi, double qj)
{
        double hold1,hold2;
        if ((qi+qj) < qs) {
                hold1 = exp(1/qs)
                hold2 = exp(1/(qe[i]+qe[j]-qs));
                return hold1*hold2;
        }
        else
                return 0;
}

/* ----------------------------------------------------------------------
  allocate arrays and map
------------------------------------------------------------------------ */

void PairSWBKS::allocate()
{
  allocated = 1;
  int n = atom->ntypes;
  map = new int[n+1];
  qset = new int[n+1];

  setflag = memory->create_2d_int_array(n+1,n+1,"pair:setflag");
  for (int i = 1; i <= n; i++)
    for (int j = i; j <= n; j++)
      setflag[i][j] = 0;

  cutsq = memory->create_2d_double_array(n+1,n+1,"pair:cutsq");

  cut_lj = memory->create_2d_double_array(n+1,n+1,"pair:cut_lj");
  cut_ljsq = memory->create_2d_double_array(n+1,n+1,"pair:cut_ljsq");
  a = memory->create_2d_double_array(n+1,n+1,"pair:a");
  rho = memory->create_2d_double_array(n+1,n+1,"pair:rho");
  c = memory->create_2d_double_array(n+1,n+1,"pair:c");
  rhoinv = memory->create_2d_double_array(n+1,n+1,"pair:rhoinv");
  buck1 = memory->create_2d_double_array(n+1,n+1,"pair:buck1");
  buck2 = memory->create_2d_double_array(n+1,n+1,"pair:buck2");
  offset = memory->create_2d_double_array(n+1,n+1,"pair:offset");

}


/* ----------------------------------------------------------------------
```

```
  global settings
------------------------------------------------------------------------- */

void PairSWBKS::settings(int narg, char **arg)
{
  if (narg < 1 || narg > 2) error->all("Illegal pair_style command");

  cut_lj_global = force->numeric(arg[0]);
  if (narg == 1) cut_coul = cut_lj_global;
  else cut_coul = force->numeric(arg[1]);

  // reset cutoffs that have been explicitly set

  if (allocated) {
    int i,j;
    for (i = 1; i <= atom->ntypes; i++)
      for (j = i+1; j <= atom->ntypes; j++)
            if (setflag[i][j]) cut_lj[i][j] = cut_lj_global;
  }
}

/* ----------------------------------------------------------------------
  set coeffs for one or more type pairs
------------------------------------------------------------------------- */

void PairSW::coeff(int narg, char **arg)
{
  int i,j,n;

  if (!allocated) allocate();

  if (narg != 3 + atom->ntypes)
    error->all("Incorrect args for pair coefficients");

  // insure I,J args are * *

  if (strcmp(arg[0],"*") != 0 || strcmp(arg[1],"*") != 0)
    error->all("Incorrect args for pair coefficients");

  // read args that map atom types to elements in potential file
  // map[i] = which element the Ith atom type is, -1 if NULL
  // nelements = # of unique elements
  // elements = list of element names

  if (elements) {
    for (i = 0; i < nelements; i++) delete [] elements[i];
    delete [] elements;
  }
  elements = new char*[atom->ntypes];
  for (i = 0; i < atom->ntypes; i++) elements[i] = NULL;

  nelements = 0;
  for (i = 3; i < narg; i++) {
    for (j = 0; j < nelements; j++)
      if (strcmp(arg[i],elements[j]) == 0) break;
```

```
    map[i-2] = j;
    if (j == nelements) {
      n = strlen(arg[i]) + 1;
      elements[j] = new char[n];
      strcpy(elements[j],arg[i]);
      nelements++;
    }
  }

  // read potential file and initialize potential parameters

  read_file(arg[2]);
  setup();

  // clear setflag since coeff() called once with I,J = * *

  n = atom->ntypes;
  for (int i = 1; i <= n; i++)
    for (int j = i; j <= n; j++)
      setflag[i][j] = 0;

  // set setflag i,j for type pairs where both are mapped to elements

  int count = 0;
  for (int i = 1; i <= n; i++)
    for (int j = i; j <= n; j++)
      if (map[i] >= 0 && map[j] >= 0) {
            setflag[i][j] = 1;
            count++;
      }

  if (count == 0) error->all("Incorrect args for pair coefficients");
}
/* ----------------------------------------------------------------------
   init specific to this pair style
------------------------------------------------------------------------- */

void PairSWBKS::init_style()
{
  if (atom->tag_enable == 0)
    error->all("Pair style Stillinger-Weber requires atom IDs");
  if (force->newton_pair == 0)
    error->all("Pair style Stillinger-Weber requires newton pair on");

  cut_coulsq = cut_coul * cut_coul;

  // insure use of KSpace long-range solver, set g_ewald for buck/coul/long

  if (force->kspace == NULL)
    error->all("Pair style is incompatible with KSpace style");
  g_ewald = force->kspace->g_ewald;

  // need a full neighbor list
  int irequest = neighbor->request(this);
  neighbor->requests[irequest]->half = 0;
```

```
  neighbor->requests[irequest]->full = 1;
}

/* ----------------------------------------------------------------------
   init for one type pair i,j and corresponding j,i
------------------------------------------------------------------------- */

double PairSW::init_one(int i, int j)
{
  if (setflag[i][j] == 0) error->all("All pair coeffs are not set");

  double cuta = MAX(cut_lj[i][j],cut_coul);
  double cut = MAX(cuta,cutmax);
  cut_ljsq[i][j] = cut_lj[i][j] * cut_lj[i][j];

  rhoinv[i][j] = 1.0/rho[i][j];
  buck1[i][j] = a[i][j]/rho[i][j];
  buck2[i][j] = 6.0*c[i][j];

  if (offset_flag) {
    double rexp = exp(-cut_lj[i][j]/rho[i][j]);
    offset[i][j] = a[i][j]*rexp - c[i][j]/pow(cut_lj[i][j],6.0);
  } else offset[i][j] = 0.0;

  cut_ljsq[j][i] = cut_ljsq[i][j];
  a[j][i] = a[i][j];
  c[j][i] = c[i][j];
  rhoinv[j][i] = rhoinv[i][j];
  buck1[j][i] = buck1[i][j];
  buck2[j][i] = buck2[i][j];
  offset[j][i] = offset[i][j];
  swcheck[j][i] = swcheck[i][j];

  return cut;
}

/* ---------------------------------------------------------------------- */

void PairSW::read_file(char *file)
{
        int params_per_line = 13;
        int param_type=0;
        char **words = new char*[params_per_line+1];
        int hold,n,nwords,ielement,jelement,swset,mixset;
        char line[MAXLINE],*ptr,look;
        int eof = 0;
  // open file on proc 0
        n = atom->ntypes;
        for (int i = 1; i <= n; i++) {
                qset[n] = 0;
                for (int j = i; j <= n; j++)
                        setflag[i][j] = 0;
        }
        swset = mixset = 0;
```

```
        FILE *fp;
        if (comm->me == 0) {
                fp = fopen(file,"r");
                if (fp == NULL) {
                        char str[128];
                        sprintf(str,"Cannot open Stillinger-Weber potential file %s",file);
                        error->one(str);
                }
        }

  // read each set of params from potential file
  // one set of params can span multiple lines

        while (1) {
                param_type=0;
                if (comm->me == 0) {
                        ptr = fgets(line,MAXLINE,fp);
                        if (ptr == NULL) {
                                eof = 1;
                                fclose(fp);
                        } else n = strlen(line) + 1;
                }
                MPI_Bcast(&eof,1,MPI_INT,0,world);
                if (eof) break;
                MPI_Bcast(&n,1,MPI_INT,0,world);
                MPI_Bcast(line,n,MPI_CHAR,0,world);

   // strip comment, skip line if blank

                if (ptr = strchr(line,'#')) *ptr = '\0';
                nwords = atom->count_words(line);
                if (nwords == 0) continue;

                // determine which parameter type line consists of

                if (ptr=strchr(line,'S')) {
                        look = *(ptr+1);
                        if (look == 'W') {
                                params_per_line = 13;
                                param_type = 1;
                        }
                }
                if (ptr=strchr(line,'Q')) {
                        if (param_type) error->all("Incorrect format in SWBKS potential file");
                        params_per_line = 3;
                        param_type = 2;
                }
                if (ptr=strchr(line,'M')) {
                        look = *(ptr+1);
                        if (look == 'I') {
                                look = *(ptr+2);
                                if (look == 'X') {
                                        if (param_type) error->all("Incorrect format in SWBKS
potential file");
                                        params_per_line = 5;
```

```
                                        param_type = 3;
                                }
                        }
                }
                if (param_type == 0) {
                        params_per_line = 5;
                        param_type = 4;
                }

        // concatenate additional lines until have params_per_line words

          while (nwords < params_per_line) {
                        n = strlen(line);
                        if (comm->me == 0) {
                                ptr = fgets(&line[n],MAXLINE-n,fp);
                                if (ptr == NULL) {
                                        eof = 1;
                                        fclose(fp);
                                } else n = strlen(line) + 1;
                        }
                        MPI_Bcast(&eof,1,MPI_INT,0,world);
                        if (eof) break;
                        MPI_Bcast(&n,1,MPI_INT,0,world);
                        MPI_Bcast(line,n,MPI_CHAR,0,world);
                        if (ptr = strchr(line,'#')) *ptr = '\0';
                        nwords = atom->count_words(line);
                }

                if (nwords != params_per_line) error->all("Incorrect format in SWBKS potential file");

    // words = ptrs to all words in line

          nwords = 0;
          words[nwords++] = strtok(line," \t\n\r\f");
          while (words[nwords++] = strtok(NULL," \t\n\r\f")) continue;

        // assign parameters according to their line style

        //Stillinger-Weber settings
                if (param_type == 1) {
                        cout << "Stillinger-Weber line" << endl;
                        if (swset == 1) error->all("Duplicate SW parameters not allowed");
                        swset = 1;
                        for (ielement = 0; ielement < nelements; ielement++)
                                if (strcmp(words[1],elements[ielement]) == 0) break;
                        if (ielement == nelements) error->all("Atom type for SW parameters not
consistent");

                        swparams.element = ielement;
                        swparams.epsilon = atof(words[2]);
                        swparams.sigma = atof(words[3]);
                        swparams.littlea = atof(words[4]);
                        swparams.lambda = atof(words[5]);
                        swparams.gamma = atof(words[6]);
                        swparams.costheta = atof(words[7]);
```

```
                swparams.biga = atof(words[8]);
                swparams.bigb = atof(words[9]);
                swparams.powerp = atof(words[10]);
                swparams.powerq = atof(words[11]);
                swparams.tol = atof(words[12]);

                if (swparams.epsilon < 0.0 || swparams.sigma < 0.0 ||
                swparams.littlea < 0.0 || swparams.lambda < 0.0 ||
                swparams.gamma < 0.0 || swparams.biga < 0.0 ||
                swparams.bigb < 0.0 || swparams.powerp < 0.0 ||
                swparams.powerq < 0.0 || swparams.tol < 0.0)
                        error->all("Illegal Stillinger-Weber parameter");

        //Ionic charge settings
        } else if (param_type == 2) {
                for (ielement = 0; ielement < nelements; ielement++)
                        if (strcmp(words[1],elements[ielement]) == 0) break;
                if (ielement == nelements) error->all("Atom type for charge parameters not
consistent");
                if (qset[ielement+1] == 1) error->all("Duplicate charge parameters found");
                q[ielement] = atof(words[2]);
                qset[ielement+1]=1;

        //Jiang and Brown hybrid settings
        } else if (param_type == 3) {
                if (mixset == 1) error->all("Duplicate SWBKS parameters found");
                rs = atof(words[1]);
                ro = atof(words[2]);
                qs = atof(words[3]);
                eo = atof(words[4]);
                mixset = 1;

        //buck/coul/long (BKS) settings
        } else if (param_type == 4) {
                for (ielement = 0; ielement < nelements; ielement++)
                        if (strcmp(words[0],elements[ielement]) == 0) break;
                if (ielement == nelements) error->all("Atom type for buck/coul/long
parameters not consistent");
                for (jelement = 0; jelement < nelements; jelement++)
                        if (strcmp(words[1],elements[jelement]) == 0) break;
                if (jelement == nelements) error->all("Atom type for buck/coul/long
parameters not consistent");
                a[ielement][jelement] = atof(words[2]);
                rho[ielement][jelement] = atof(words[3]);

                if (rho[ielement][jelement] <= 0) error->all("Incorrect args for buck/coul/long
pair coefficients");

                c[ielement][jelement] = atof(words[4]);
                cut_lj[ielement][jelement] = cut_lj_global;
                if (setflag[ielement+1][jelement+1] == 1) error->all("Duplicate buck/coul/long
parameters found");
                setflag[ielement+1][jelement+1] = 1;
                cout << a[ielement][jelement] << " " << rho[ielement][jelement] << " " <<
c[ielement][jelement] << endl;
```

```
                    }
            }
  delete [] words;
}

/* ---------------------------------------------------------------------- */

void PairSW::setup()
{
  int i,j,k,m,n;
  double rtmp;


  // compute parameter values derived from inputs

  // set cutsq using shortcut to reduce neighbor list for accelerated
  // calculations. cut must remain unchanged as it is a potential parameter
  // (cut = a*sigma)

    swparams.cut = swparams.sigma*swparams.littlea;

    rtmp = swparams.cut;
    if (swparams.tol > 0.0) {
      if (swparams.tol > 0.01) swparams.tol = 0.01;
      if (swparams.gamma < 1.0)
        rtmp = rtmp +
          swparams.gamma * swparams.sigma / log(swparams.tol);
      else rtmp = rtmp +
                swparams.sigma / log(swparams.tol);
    }
    swparams.cutsq = rtmp * rtmp;

    swparams.sigma_gamma = swparams.sigma*swparams.gamma;
    swparams.lambda_epsilon = swparams.lambda*swparams.epsilon;
    swparams.lambda_epsilon2 = 2.0*swparams.lambda*swparams.epsilon;
    swparams.c1 = swparams.biga*swparams.epsilon *
      swparams.powerp*swparams.bigb *
      pow(swparams.sigma,swparams.powerp);
    swparams.c2 = swparams.biga*swparams.epsilon*swparams.powerq *
      pow(swparams.sigma,swparams.powerq);
    swparams.c3 = swparams.biga*swparams.epsilon*swparams.bigb *
      pow(swparams.sigma,swparams.powerp+1.0);
    swparams.c4 = swparams.biga*swparams.epsilon *
      pow(swparams.sigma,swparams.powerq+1.0);
    swparams.c5 = swparams.biga*swparams.epsilon*swparams.bigb *
      pow(swparams.sigma,swparams.powerp);
    swparams.c6 = swparams.biga*swparams.epsilon *
      pow(swparams.sigma,swparams.powerq);


  // set cutmax to max of all params

  cutmax = sqrt(swparams.cutsq);
}
```

```
/* ---------------------------------------------------------------------- */

void PairSW::twobody(Param *param, double rsq, double &fforce,
                     int eflag, double &eng)
{
  double r,rinvsq,rp,rq,rainv,rainvsq,expsrainv;

  r = sqrt(rsq);
  rinvsq = 1.0/rsq;
  rp = pow(r,-param->powerp);
  rq = pow(r,-param->powerq);
  rainv = 1.0 / (r - param->cut);
  rainvsq = rainv*rainv*r;
  expsrainv = exp(param->sigma * rainv);
  fforce = (param->c1*rp - param->c2*rq +
            (param->c3*rp -param->c4*rq) * rainvsq) * expsrainv * rinvsq;
  if (eflag) eng = (param->c5*rp - param->c6*rq) * expsrainv;
}

/* ---------------------------------------------------------------------- */

void PairSW::threebody(Param *params, double rsq1, double rsq2,
                       double *delr1, double *delr2,
                       double *fj, double *fk, int eflag, double &eng)
{
  double r1,rinvsq1,rainv1,gsrainv1,gsrainvsq1,expgsrainv1;
  double r2,rinvsq2,rainv2,gsrainv2,gsrainvsq2,expgsrainv2;
  double rinv12,cs,delcs,delcssq,facexp,facrad,frad1,frad2;
  double facang,facang12,csfacang,csfac1,csfac2;

  r1 = sqrt(rsq1);
  rinvsq1 = 1.0/rsq1;
  rainv1 = 1.0/(r1 - params->cut);
  gsrainv1 = params->sigma_gamma * rainv1;
  gsrainvsq1 = gsrainv1*rainv1/r1;
  expgsrainv1 = exp(gsrainv1);

  r2 = sqrt(rsq2);
  rinvsq2 = 1.0/rsq2;
  rainv2 = 1.0/(r2 - params->cut);
  gsrainv2 = params->sigma_gamma * rainv2;
  gsrainvsq2 = gsrainv2*rainv2/r2;
  expgsrainv2 = exp(gsrainv2);

  rinv12 = 1.0/(r1*r2);
  cs = (delr1[0]*delr2[0] + delr1[1]*delr2[1] + delr1[2]*delr2[2]) * rinv12;
  delcs = cs - params->costheta;
  delcssq = delcs*delcs;

  facexp = expgsrainv1*expgsrainv2;

  // facrad = sqrt(paramij->lambda_epsilon*paramik->lambda_epsilon) *
  //          facexp*delcssq;
```

```
  facrad = params->lambda_epsilon * facexp*delcssq;
  frad1 = facrad*gsrainvsq1;
  frad2 = facrad*gsrainvsq2;
  facang = params->lambda_epsilon2 * facexp*delcs;
  facang12 = rinv12*facang;
  csfacang = cs*facang;
  csfac1 = rinvsq1*csfacang;

  fj[0] = delr1[0]*(frad1+csfac1)-delr2[0]*facang12;
  fj[1] = delr1[1]*(frad1+csfac1)-delr2[1]*facang12;
  fj[2] = delr1[2]*(frad1+csfac1)-delr2[2]*facang12;

  csfac2 = rinvsq2*csfacang;

  fk[0] = delr2[0]*(frad2+csfac2)-delr1[0]*facang12;
  fk[1] = delr2[1]*(frad2+csfac2)-delr1[1]*facang12;
  fk[2] = delr2[2]*(frad2+csfac2)-delr1[2]*facang12;

  if (eflag) eng = facrad;
}
```

"SiO.swbks"

```
# Needs Stillinger-Weber parameters for 1 element, BKS (buck/coul/long)
# parameters for all element pairs, ideal charges for all elements and mixing terms.

# Begin Stillinger-Weber line with sw followed by the element.
# Begin BKS lines with both element names.
# Begin charge lines with Q followed by the element.
# Begin charge mixing line with MIX.

# LAMMPS reads all values in "metal" units. Note that units from papers differ

# SW format: SW element epsilon(eV) sigma(A) a lambda gamma costheta0 A B p q
tol
# BKS format: element1 element2 A(eV) b(1/A) c(eV*A^6)
# Charge format: Q element qnaught(electron charge)
# Mix format:  MIX rs(A) ro(A) qs(electron charge) eo(eV)

Si Si 0 0 0
SW Si 2.1683 2.0951 1.80 21.0 1.20 -0.333333333333 7.049556277 0.6022245584 4.0
0.0 0.0
Si O 18003.7572 4.87318 133.5381
O O 1388.7730 2.76000 175.0000
Q Si 2.4
Q O -1.2
MIX 3.14265 2.51412 2.727491 58.541539891
```

"pair_sw_wfnho.h"

```
/* ----------------------------------------------------------------------
   LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
   http://lammps.sandia.gov, Sandia National Laboratories
   Steve Plimpton, sjplimp@sandia.gov

   Copyright (2003) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level LAMMPS directory.
------------------------------------------------------------------------- */

#ifdef PAIR_CLASS

PairStyle(sw/wfnho,PairSWWFNHO)

#else

#ifndef LMP_PAIR_SW_WFNHO_H
#define LMP_PAIR_SW_WFNHO_H

#include "pair.h"
#define PIVAL 3.1415926535898

namespace LAMMPS_NS {

class PairSWWFNHO : public Pair {
 public:
  PairSWWFNHO(class LAMMPS *);
  ~PairSWWFNHO();
  void compute(int, int);
  void settings(int, char **);
  void coeff(int, char **);
  double init_one(int, int);
  void init_style();

 private:
  struct Param {
    double epsilon,sigma;
    double aij,aik,cij,lambda,gammaij,gammaik,costheta;
    double biga,bigb;
    double powerp,powerq;
    double tol;
    double cutpair,cutij,cutik,cutpairsq,cutijsq,cutiksq;
    double sigma_gammaij,sigma_gammaik,lambda_epsilon,lambda_epsilon2;
    double c1,c2,c3,c4,c5,c6;
    int ielement,jelement,kelement;
  };
   struct Softparam {
    double ma,mb,mc,md,me,bigr,bigd;
```

```
  int ielement,jelement;
 };
 double cutmax;               // max cutoff for all elements
 int nelements;              // # of unique elements
 char **elements;            // names of unique elements
 int ***elem2param;          // mapping from element triplets to parameters
 int **elem2soft;                    // mapping from element doubles to bond softening
 int *map;                  // mapping from atom types to elements
 int nparams;               // # of stored parameter sets
 int maxparam;              // max # of parameter sets
 Param *params;             // parameter set for an I-J-K interaction
 int nsofts;                         // # of bond softening parameter sets
 int maxsofts;                       // max # of softening parameter sets
 Softparam *soft;           // parameter set for the bond softening
 int **softflag;
 double *coord;
 void allocate();
 void read_file(char *);
 void setup();
 void twobody(Param *, double, double &, int, double &);
 void threebody(Param *, double, double, double *, double *,
                double *, double *, int, double &);
 double gsoft(double, int, int);
};

}

#endif
#endif
```

"pair_sw_wfnho.cpp"

```
/* ----------------------------------------------------------------------
   LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
   http://lammps.sandia.gov, Sandia National Laboratories
   Steve Plimpton, sjplimp@sandia.gov

   Copyright (2003) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level LAMMPS directory.
------------------------------------------------------------------------- */

/* ----------------------------------------------------------------------
   Contributing author: Lucas Hale
   Modified from the Stillinger-Weber potential by: Aidan Thompson (SNL)
------------------------------------------------------------------------- */

#include "math.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "pair_sw_wfnho.h"
#include "atom.h"
#include "neighbor.h"
#include "neigh_request.h"
#include "force.h"
#include "comm.h"
#include "memory.h"
#include "neighbor.h"
#include "neigh_list.h"
#include "memory.h"
#include "error.h"

using namespace LAMMPS_NS;

#define MAXLINE 1024
#define DELTA 4

/* ---------------------------------------------------------------------- */

PairSWWFNHO::PairSWWFNHO(LAMMPS *lmp) : Pair(lmp)
{
  single_enable = 0;
  one_coeff = 1;

  nelements = 0;
  elements = NULL;
  nparams = maxparam = 0;
  nsofts = 0;
  params = NULL;
```

```
  elem2param = NULL;
  soft = NULL;
  elem2soft = NULL;
}

/* ----------------------------------------------------------------------
   check if allocated, since class can be destructed when incomplete
------------------------------------------------------------------------- */

PairSWWFNHO::~PairSWWFNHO()
{
  if (elements)
    for (int i = 0; i < nelements; i++) delete [] elements[i];
  delete [] elements;
  memory->sfree(params);
  memory->sfree(soft);
  memory->destroy_3d_int_array(elem2param);

  if (allocated) {
    memory->destroy_2d_int_array(setflag);
    memory->destroy_2d_int_array(softflag);
    memory->destroy_2d_double_array(cutsq);
    delete [] map;
    delete [] coord;
  }
}

/* ---------------------------------------------------------------------- */

void PairSWWFNHO::compute(int eflag, int vflag)
{
  int i,j,k,ii,jj,kk,inum,jnum,jnumm1,itag,jtag;
  int itype,jtype,ktype,ijparam,ijkparam;
  double xtmp,ytmp,ztmp,delx,dely,delz,evdwl,fpair;
  double rsq,rsq1,rsq2,bigr,bigd,gij,r;
  double delr1[3],delr2[3],fj[3],fk[3];
  int *ilist,*jlist,*numneigh,**firstneigh;

  evdwl = 0.0;
  if (eflag || vflag) ev_setup(eflag,vflag);
  else evflag = vflag_fdotr = 0;

  double **x = atom->x;
  double **f = atom->f;
  int *tag = atom->tag;
  int *type = atom->type;
  int nlocal = atom->nlocal;
  int newton_pair = force->newton_pair;

  inum = list->inum;
  ilist = list->ilist;
  numneigh = list->numneigh;
  firstneigh = list->firstneigh;

  // calculate coordination number for softening function
```

```
for (ii = 0; ii < inum; ii++)
{
  i = ilist[ii];
  coord[tag[i]]=0;
}
for (ii = 0; ii < inum; ii++) {
  i = ilist[ii];
  itag = tag[i];
  itype = map[type[i]];
  xtmp = x[i][0];
  ytmp = x[i][1];
  ztmp = x[i][2];

  jlist = firstneigh[i];
  jnum = numneigh[i];
  for (jj = 0; jj < jnum; jj++) {
    j=jlist[jj];
    jtag = tag[j];
    jtype = map[type[j]];

    if (softflag[itype][jtype])
          {
            ijparam = elem2param[itype][jtype][jtype];
            bigr = soft[elem2soft[itype][jtype]].bigr;
            bigd = soft[elem2soft[itype][jtype]].bigd;
            delx = xtmp - x[j][0];
            dely = ytmp - x[j][1];
            delz = ztmp - x[j][2];
            rsq = delx*delx + dely*dely + delz*delz;
            r = sqrt(rsq)/params[ijparam].sigma;

            if (r < (bigr - bigd)) coord[itag] += 1;
            else if (r < (bigr + bigd))
              coord[itag] += 1 - (r-bigr+bigd)/(2*bigd) + sin(PIVAL*(r-bigr+bigd)/bigd)/(2*PIVAL);
          }
  }
}


// loop over full neighbor list of my atoms

for (ii = 0; ii < inum; ii++) {
  i = ilist[ii];
  itag = tag[i];
  itype = map[type[i]];
  xtmp = x[i][0];
  ytmp = x[i][1];
  ztmp = x[i][2];

  // two-body interactions, skip half of them

  jlist = firstneigh[i];
  jnum = numneigh[i];

  for (jj = 0; jj < jnum; jj++) {
```

```
  j = jlist[jj];
  jtag = tag[j];

  if (itag > jtag) {
        if ((itag+jtag) % 2 == 0) continue;
  } else if (itag < jtag) {
        if ((itag+jtag) % 2 == 1) continue;
  } else {
        if (x[j][2] < ztmp) continue;
        if (x[j][2] == ztmp && x[j][1] < ytmp) continue;
        if (x[j][2] == ztmp && x[j][1] == ytmp && x[j][0] < xtmp) continue;
  }

  jtype = map[type[j]];

  delx = xtmp - x[j][0];
  dely = ytmp - x[j][1];
  delz = ztmp - x[j][2];
  rsq = delx*delx + dely*dely + delz*delz;

  ijparam = elem2param[itype][jtype][jtype];
  if (rsq > params[ijparam].cutpairsq) continue;

  twobody(&params[ijparam],rsq,fpair,eflag,evdwl);

        if (softflag[itype][jtype]) gij = gsoft(coord[itag],itype,jtype);
        else if (softflag[jtype][itype]) gij = gsoft(coord[jtag],jtype,itype);
        else gij = 1;
        evdwl = gij*evdwl;
        fpair = gij*fpair;

  f[i][0] += delx*fpair;
  f[i][1] += dely*fpair;
  f[i][2] += delz*fpair;
  f[j][0] -= delx*fpair;
  f[j][1] -= dely*fpair;
  f[j][2] -= delz*fpair;

  if (evflag) ev_tally(i,j,nlocal,newton_pair,
                                  evdwl,0.0,fpair,delx,dely,delz);
}

jnumm1 = jnum - 1;

for (jj = 0; jj < jnumm1; jj++) {
  j = jlist[jj];
  jtype = map[type[j]];
  delr1[0] = x[j][0] - xtmp;
  delr1[1] = x[j][1] - ytmp;
  delr1[2] = x[j][2] - ztmp;
  rsq1 = delr1[0]*delr1[0] + delr1[1]*delr1[1] + delr1[2]*delr1[2];

  for (kk = jj+1; kk < jnum; kk++) {
        k = jlist[kk];
        ktype = map[type[k]];
```

```
            ijkparam = elem2param[itype][jtype][ktype];
            if (rsq1 > params[ijkparam].cutijsq) continue;

            delr2[0] = x[k][0] - xtmp;
            delr2[1] = x[k][1] - ytmp;
            delr2[2] = x[k][2] - ztmp;
            rsq2 = delr2[0]*delr2[0] + delr2[1]*delr2[1] + delr2[2]*delr2[2];
            if (rsq2 > params[ijkparam].cutiksq) continue;

            threebody(&params[ijkparam],rsq1,rsq2,delr1,delr2,fj,fk,eflag,evdwl);

            f[i][0] -= fj[0] + fk[0];
            f[i][1] -= fj[1] + fk[1];
            f[i][2] -= fj[2] + fk[2];
            f[j][0] += fj[0];
            f[j][1] += fj[1];
            f[j][2] += fj[2];
            f[k][0] += fk[0];
            f[k][1] += fk[1];
            f[k][2] += fk[2];

            if (evflag) ev_tally3(i,j,k,evdwl,0.0,fj,fk,delr1,delr2);
    }
   }
  }
  if (vflag_fdotr) virial_compute();

}

/* ---------------------------------------------------------------------- */
double PairSWWFNHO::gsoft(double cn, int i, int j)
{
            double ma,mb,mc,md,me,first,second;
            int ij = elem2soft[i][j];

            ma = soft[ij].ma;
            mb = soft[ij].mb;
            mc = soft[ij].mc;
            md = soft[ij].md;
            me = soft[ij].me;
            first = ma/(exp((mb-cn)/mc)+1);
            second = exp(md*(cn-me)*(cn-me));

            return first*second;
}

/* ---------------------------------------------------------------------- */

void PairSWWFNHO::allocate()
{
  allocated = 1;
  int n = atom->ntypes;
  int num = ceil(atom->natoms);

  coord = new double[num+1];
```

```
  setflag = memory->create_2d_int_array(n+1,n+1,"pair:setflag");
  cutsq = memory->create_2d_double_array(n+1,n+1,"pair:cutsq");
  softflag = memory->create_2d_int_array(n+1,n+1,"pair:softflag");
  map = new int[n+1];
}

/* ----------------------------------------------------------------------
   global settings
------------------------------------------------------------------------- */

void PairSWWFNHO::settings(int narg, char **arg)
{
  if (narg != 0) error->all("Illegal pair_style command");
}

/* ----------------------------------------------------------------------
   set coeffs for one or more type pairs
------------------------------------------------------------------------- */

void PairSWWFNHO::coeff(int narg, char **arg)
{
  int i,j,n;

  if (!allocated) allocate();

  if (narg != 3 + atom->ntypes)
    error->all("Incorrect args for pair coefficients");

  // insure I,J args are * *

  if (strcmp(arg[0],"*") != 0 || strcmp(arg[1],"*") != 0)
    error->all("Incorrect args for pair coefficients");

  // read args that map atom types to elements in potential file
  // map[i] = which element the Ith atom type is, -1 if NULL
  // nelements = # of unique elements
  // elements = list of element names

  if (elements) {
    for (i = 0; i < nelements; i++) delete [] elements[i];
    delete [] elements;
  }
  elements = new char*[atom->ntypes];
  for (i = 0; i < atom->ntypes; i++) elements[i] = NULL;

  nelements = 0;
  for (i = 3; i < narg; i++) {
    if (strcmp(arg[i],"NULL") == 0) {
      map[i-2] = -1;
      continue;
    }
    for (j = 0; j < nelements; j++)
      if (strcmp(arg[i],elements[j]) == 0) break;
    map[i-2] = j;
    if (j == nelements) {
```

```
      n = strlen(arg[i]) + 1;
      elements[j] = new char[n];
      strcpy(elements[j],arg[i]);
      nelements++;
    }
  }

  // read potential file and initialize potential parameters

  read_file(arg[2]);
   setup();

  // clear setflag since coeff() called once with I,J = * *

  n = atom->ntypes;
  for (int i = 1; i <= n; i++)
    for (int j = i; j <= n; j++)
      setflag[i][j] = 0;

  // set setflag i,j for type pairs where both are mapped to elements

  int count = 0;
  for (int i = 1; i <= n; i++)
    for (int j = i; j <= n; j++)
      if (map[i] >= 0 && map[j] >= 0) {
            setflag[i][j] = 1;
            count++;
      }

  if (count == 0) error->all("Incorrect args for pair coefficients");
}

/* ----------------------------------------------------------------------
   init specific to this pair style
------------------------------------------------------------------------- */

void PairSWWFNHO::init_style()
{
  if (atom->tag_enable == 0)
    error->all("Pair style Stillinger-Weber requires atom IDs");
  if (force->newton_pair == 0)
    error->all("Pair style Stillinger-Weber requires newton pair on");

  // need a full neighbor list

  int irequest = neighbor->request(this);
  neighbor->requests[irequest]->half = 0;
  neighbor->requests[irequest]->full = 1;
}

/* ----------------------------------------------------------------------
   init for one type pair i,j and corresponding j,i
------------------------------------------------------------------------- */

double PairSWWFNHO::init_one(int i, int j)
```

```
{
  if (setflag[i][j] == 0) error->all("All pair coeffs are not set");

  return cutmax;
}

/* ---------------------------------------------------------------------- */

void PairSWWFNHO::read_file(char *file)
{
  int params_per_line = 17;
  int soft_per_line = 10;
  int per_line;
  int param_type = 0;
  char **words = new char*[params_per_line+1];
  char look1,look2,look3;

  memory->sfree(params);
  params = NULL;
  nparams = maxparam = 0;
  if (soft !=NULL) free(soft);
  soft = NULL;
  nsofts = maxsofts = 0;
  for (int y=0;y<atom->ntypes;y++)
    for (int z=0;z<atom->ntypes;z++)
      softflag[y][z]=0;

  // open file on proc 0

  FILE *fp;
  if (comm->me == 0) {
    fp = fopen(file,"r");
    if (fp == NULL) {
      char str[128];
      sprintf(str,"Cannot open Stillinger-Weber potential file %s",file);
      error->one(str);
    }
  }

  // read each set of params from potential file
  // one set of params can span multiple lines
  // store params if all 3 element tags are in element list

  int n,nwords,ielement,jelement,kelement;
  char line[MAXLINE],*ptr;
  int eof = 0;

  while (1) {
    param_type=0;
    per_line = params_per_line;
    if (comm->me == 0) {
      ptr = fgets(line,MAXLINE,fp);
      if (ptr == NULL) {
              eof = 1;
              fclose(fp);
```

```
  } else n = strlen(line) + 1;
}
MPI_Bcast(&eof,1,MPI_INT,0,world);
if (eof) break;
MPI_Bcast(&n,1,MPI_INT,0,world);
MPI_Bcast(line,n,MPI_CHAR,0,world);

// strip comment, skip line if blank

if (ptr = strchr(line,'#')) *ptr = '\0';
nwords = atom->count_words(line);
if (nwords == 0) continue;
// determine which parameter type line consists of

if ((ptr=strchr(line,'S'))) {
        look1 = *(ptr+1);
  look2 = *(ptr+2);
        look3 = *(ptr+3);
        if (look1 == 'o' && look2 == 'f' && look3 == 't') {
          per_line = soft_per_line;
                param_type = 1;
        }
}

while (nwords < per_line) {
  n = strlen(line);
  if (comm->me == 0) {
    ptr = fgets(&line[n],MAXLINE-n,fp);
    if (ptr == NULL) {
            eof = 1;
            fclose(fp);
    } else n = strlen(line) + 1;
  }
  MPI_Bcast(&eof,1,MPI_INT,0,world);
  if (eof) break;
  MPI_Bcast(&n,1,MPI_INT,0,world);
  MPI_Bcast(line,n,MPI_CHAR,0,world);
  if (ptr = strchr(line,'#')) *ptr = '\0';
  nwords = atom->count_words(line);
}

if (nwords != per_line) error->all("Incorrect format in Stillinger-Weber potential file");

// words = ptrs to all words in line

nwords = 0;
words[nwords++] = strtok(line," \t\n\r\f");
while (words[nwords++] = strtok(NULL," \t\n\r\f")) continue;

//2 and 3 body parameter settings
if (param_type == 0) {
// ielement,jelement,kelement = 1st args
// if all 3 args are in element list, then parse this line
// else skip to next entry in file
```

```
  for (ielement = 0; ielement < nelements; ielement++)
    if (strcmp(words[0],elements[ielement]) == 0) break;
  if (ielement == nelements) continue;
  for (jelement = 0; jelement < nelements; jelement++)
    if (strcmp(words[1],elements[jelement]) == 0) break;
  if (jelement == nelements) continue;
  for (kelement = 0; kelement < nelements; kelement++)
    if (strcmp(words[2],elements[kelement]) == 0) break;
  if (kelement == nelements) continue;

  // load up parameter settings and error check their values

  if (nparams == maxparam) {
    maxparam += DELTA;
    params = (Param *) memory->srealloc(params,maxparam*sizeof(Param),
                                        "pair:params");
  }

  params[nparams].ielement = ielement;
  params[nparams].jelement = jelement;
  params[nparams].kelement = kelement;
  params[nparams].epsilon = atof(words[3]);
  params[nparams].sigma = atof(words[4]);
  params[nparams].aij = atof(words[5]);
  params[nparams].aik = atof(words[6]);
  params[nparams].lambda = atof(words[7]);
  params[nparams].gammaij = atof(words[8]);
  params[nparams].gammaik = atof(words[9]);
  params[nparams].costheta = atof(words[10]);
  params[nparams].biga = atof(words[11]);
  params[nparams].bigb = atof(words[12]);
  params[nparams].powerp = atof(words[13]);
  params[nparams].powerq = atof(words[14]);
  params[nparams].cij = atof(words[15]);
  params[nparams].tol = atof(words[16]);
  /*
  if (params[nparams].epsilon < 0.0 || params[nparams].sigma < 0.0 ||
          params[nparams].aij < 0.0 || params[nparams].aik < 0.0 ||
          params[nparams].lambda < 0.0 || params[nparams].gammaij < 0.0 ||
          params[nparams].gammaik < 0.0 || ||
          params[nparams].bigb < 0.0 || params[nparams].powerp < 0.0 ||
          params[nparams].powerq < 0.0 || params[nparams].cij < 0.0 ||
          params[nparams].tol < 0.0)
    error->all("Illegal Stillinger-Weber parameter");
  */
  nparams++;
} else if (param_type == 1) {
          for (ielement = 0; ielement < nelements; ielement++)
      if (strcmp(words[1],elements[ielement]) == 0) break;
    if (ielement == nelements) continue;
    for (jelement = 0; jelement < nelements; jelement++)
      if (strcmp(words[2],elements[jelement]) == 0) break;
    if (jelement == nelements) continue;

  // load up parameter settings and error check their values
```

```
        softflag[ielement][jelement] = 1;
        if (nsofts == maxsofts) {
    maxsofts += DELTA;
    soft = (Softparam *) memory->srealloc(soft,maxsofts*sizeof(Softparam),
                                          "pair:soft");
   }
   soft[nsofts].ielement = ielement;
   soft[nsofts].jelement = jelement;
   soft[nsofts].ma = atof(words[3]);
   soft[nsofts].mb = atof(words[4]);
   soft[nsofts].mc = atof(words[5]);
   soft[nsofts].md = atof(words[6]);
   soft[nsofts].me = atof(words[7]);
   soft[nsofts].bigr = atof(words[8]);
   soft[nsofts].bigd = atof(words[9]);
   nsofts++;
  }
  }
 delete [] words;
}

/* ---------------------------------------------------------------------- */

void PairSWWFNHO::setup()
{
  int i,j,k,m,n,o,p;
  double rtmp1,rtmp2,rtmp3;

  // set elem2param for all triplet combinations
  // must be a single exact match to lines read from file
  // do not allow for ACB in place of ABC

  if (elem2param) memory->destroy_3d_int_array(elem2param);
  elem2param = memory->create_3d_int_array(nelements,nelements,nelements,
                                           "pair:elem2param");
  if (elem2soft) memory->destroy_2d_int_array(elem2soft);
  elem2soft = memory->create_2d_int_array(nelements,nelements,"pair:elem2soft");

  for (i = 0; i < nelements; i++) {
    for (j = 0; j < nelements; j++) {
      o = -1;
      for (p = 0; p < nsofts; p++) {
        if (i == soft[p].ielement && j == soft[p].jelement) {
          if (o >= 0) error->all("Potential file has duplicate entry");
          o = p;
        }
      }
      elem2soft[i][j] = o;
      for (k = 0; k < nelements; k++) {
        n = -1;
        for (m = 0; m < nparams; m++) {
          if (i == params[m].ielement && j == params[m].jelement &&
              k == params[m].kelement) {
            if (n >= 0) error->all("Potential file has duplicate entry");
            n = m;
```

```
          }
        }
        if (n < 0) error->all("Potential file is missing an entry");
        elem2param[i][j][k] = n;
    }
  }
}

// compute parameter values derived from inputs

// set cutsq using shortcut to reduce neighbor list for accelerated
// calculations. cuts must remain unchanged as it is a potential parameter
// (cut = a*sigma)

for (m = 0; m < nparams; m++) {
  params[m].cutpair = params[m].sigma*params[m].cij;
  params[m].cutij = params[m].sigma*params[m].aij;
  params[m].cutik = params[m].sigma*params[m].aik;

  rtmp1 = params[m].cutpair;
  rtmp2 = params[m].cutij;
  rtmp3 = params[m].cutik;

  if (params[m].tol > 0.0) error->all("Potential not currently set to accept tol values");

  params[m].cutpairsq = rtmp1 * rtmp1;
  params[m].cutijsq = rtmp2 * rtmp2;
  params[m].cutiksq = rtmp3 * rtmp3;

  params[m].sigma_gammaij = params[m].sigma*params[m].gammaij;
  params[m].sigma_gammaik = params[m].sigma*params[m].gammaik;
  params[m].lambda_epsilon = params[m].lambda*params[m].epsilon;
  params[m].lambda_epsilon2 = 2.0*params[m].lambda*params[m].epsilon;
  params[m].c1 = params[m].biga*params[m].epsilon *
    params[m].powerp*params[m].bigb *
    pow(params[m].sigma,params[m].powerp);
  params[m].c2 = params[m].biga*params[m].epsilon*params[m].powerq *
    pow(params[m].sigma,params[m].powerq);
  params[m].c3 = params[m].biga*params[m].epsilon*params[m].bigb *
    pow(params[m].sigma,params[m].powerp+1.0);
  params[m].c4 = params[m].biga*params[m].epsilon *
    pow(params[m].sigma,params[m].powerq+1.0);
  params[m].c5 = params[m].biga*params[m].epsilon*params[m].bigb *
    pow(params[m].sigma,params[m].powerp);
  params[m].c6 = params[m].biga*params[m].epsilon *
    pow(params[m].sigma,params[m].powerq);
}

// set cutmax to max of all params

cutmax = 0.0;
for (m = 0; m < nparams; m++) {
  rtmp1 = sqrt(params[m].cutpairsq);
  if (rtmp1 > cutmax) cutmax = rtmp1;
  rtmp2 = sqrt(params[m].cutijsq);
```

```
    if (rtmp2 > cutmax) cutmax = rtmp2;
    rtmp3 = sqrt(params[m].cutiksq);
    if (rtmp3 > cutmax) cutmax = rtmp3;
  }
}

/* ---------------------------------------------------------------------- */

void PairSWWFNHO::twobody(Param *param, double rsq, double &fforce,
                          int eflag, double &eng)
{
  double r,rinvsq,rp,rq,rainv,rainvsq,expsrainv;
  r = sqrt(rsq);
  rinvsq = 1.0/rsq;
  rp = pow(r,-param->powerp);
  rq = pow(r,-param->powerq);
  rainv = 1.0 / (r - param->cutpair);
  rainvsq = rainv*rainv*r;
  expsrainv = exp(param->sigma * rainv);
  fforce = (param->c1*rp - param->c2*rq +
            (param->c3*rp -param->c4*rq) * rainvsq) * expsrainv * rinvsq;
  if (eflag) eng = (param->c5*rp - param->c6*rq) * expsrainv;
}

/* ---------------------------------------------------------------------- */

void PairSWWFNHO::threebody(Param *paramijk,
                            double rsq1, double rsq2,
                            double *delr1, double *delr2,
                            double *fj, double *fk, int eflag, double &eng)
{
  double r1,rinvsq1,rainv1,gsrainv1,gsrainvsq1,expgsrainv1;
  double r2,rinvsq2,rainv2,gsrainv2,gsrainvsq2,expgsrainv2;
  double rinv12,cs,delcs,delcssq,facexp,facrad,frad1,frad2;
  double facang,facang12,csfacang,csfac1,csfac2;

  r1 = sqrt(rsq1);
  rinvsq1 = 1.0/rsq1;
  rainv1 = 1.0/(r1 - paramijk->cutij);
  gsrainv1 = paramijk->sigma_gammaij * rainv1;
  gsrainvsq1 = gsrainv1*rainv1/r1;
  expgsrainv1 = exp(gsrainv1);

  r2 = sqrt(rsq2);
  rinvsq2 = 1.0/rsq2;
  rainv2 = 1.0/(r2 - paramijk->cutik);
  gsrainv2 = paramijk->sigma_gammaik * rainv2;
  gsrainvsq2 = gsrainv2*rainv2/r2;
  expgsrainv2 = exp(gsrainv2);

  rinv12 = 1.0/(r1*r2);
  cs = (delr1[0]*delr2[0] + delr1[1]*delr2[1] + delr1[2]*delr2[2]) * rinv12;
  delcs = cs - paramijk->costheta;
  delcssq = delcs*delcs;
```

```
  facexp = expgsrainv1*expgsrainv2;

  // facrad = sqrt(paramijk->lambda_epsilon*paramijk->lambda_epsilon) *
  //          facexp*delcssq;

  facrad = paramijk->lambda_epsilon * facexp*delcssq;
  frad1 = facrad*gsrainvsq1;
  frad2 = facrad*gsrainvsq2;
  facang = paramijk->lambda_epsilon2 * facexp*delcs;
  facang12 = rinv12*facang;
  csfacang = cs*facang;
  csfac1 = rinvsq1*csfacang;

  fj[0] = delr1[0]*(frad1+csfac1)-delr2[0]*facang12;
  fj[1] = delr1[1]*(frad1+csfac1)-delr2[1]*facang12;
  fj[2] = delr1[2]*(frad1+csfac1)-delr2[2]*facang12;

  csfac2 = rinvsq2*csfacang;

  fk[0] = delr2[0]*(frad2+csfac2)-delr1[0]*facang12;
  fk[1] = delr2[1]*(frad2+csfac2)-delr1[1]*facang12;
  fk[2] = delr2[2]*(frad2+csfac2)-delr1[2]*facang12;

  if (eflag) eng = facrad;
}
```

"SiO.sw_wfnho"

```
# Parameters for the Stillinger-Weber like potential developed by Watanabe, et. al
# multiple entries can be added to this file, LAMMPS reads the ones it needs
# these entries are in LAMMPS "metal" units:
# epsilon = eV; sigma = Angstroms
# other quantities are unitless

# format of a single element triplet entry (one or more lines):
#   element 1, element 2, element 3, epsilon, sigma, a_ij, a_ik, lambda, gamm_ij,
gamm_ik, costheta0, Aij, Bij, pij, qij, cij, tol
#
# epsilon and sigma are scaling quantaties, a_ij through costheta0 are for 3-body
interactions
# and Aij through cij are for 2 body interactions.
# NOTE! a_ij, a_ik, and cij are the cutoff distances used by the potential.  These replace
the
# single cutoff distance, a, used in the Stillinger-Weber potential.
# Watanabe and Ganster both label cij differently, but "cij" was chosen here to
minimize confusion

# format of the bond-softening function (must start line with "Soft":
#   Soft, element 1, element 2, m1, m2, m3, m4, m5, R, D
#
# The bond-softening function is only calculated for element 1 being bonded to element
2.
# All other pairings have the function = 1

# Si = parameters from Watanabe, et. al, Jap. J. Appl. Phys., v. 38 p. L366 (1999)
#
# Si(a) = parameters from Ganster, et. al, Phys. Rev. B, v. 81 p. 045315 (2010) that
exactly
# reproduces Si Si Si of Stillinger and Weber,  Phys. Rev. B, v. 31, p. 5262, (1985)
#

#E1    E2    E3    epsil  sigma  a_ij a_ik lambda gamm_ij gamm_ik costheta0       Aij
Bij         pij     qij     cij  tol

 Si    Si    Si    2.1696 2.0951 1.80 1.80 16.404 1.0473  1.0473  -0.333333333333
7.049556277   0.6022245584 4.0     0.0     1.80 0.0

 Si    Si    O     2.1696 2.0951 1.90 1.40 10.667 1.93973 0.25    -0.333333333333
7.049556277   0.6022245584 4.0     0.0     1.80 0.0
```

```
 Si   O    Si   2.1696 2.0951 1.40 1.90 10.667 0.25    1.93973 -0.333333333333
115.364065913 0.9094442793 2.58759 2.39370 1.40 0.0

 Si   O    O    2.1696 2.0951 1.65 1.65 3.1892 0.3220  0.3220  -0.333333333333
115.364065913 0.9094442793 2.58759 2.39370 1.40 0.0

 O    Si   Si   2.1696 2.0951 1.40 1.40 2.9572 0.71773 0.71773 -0.6155238
115.364065913 0.9094442793 2.58759 2.39370 1.40 0.0

 O    Si   O    2.1696 2.0951 0.0  0.0  0.0    0.0     0.0     0.0            115.364065913
0.9094442793 2.58759 2.39370 1.40 0.0

 O    O    Si   2.1696 2.0951 0.0  0.0  0.0    0.0     0.0     0.0            -12.292427744 0.0
0.0    2.24432 1.25 0.0

 O    O    O    2.1696 2.0951 0.0  0.0  0.0    0.0     0.0     0.0            -12.292427744 0.0
0.0    2.24432 1.25 0.0

#Parameters for the bond-softening function
#Soft E1   E2   m1     m2     m3     m4     m5     R   D
 Soft O    Si   0.0970 1.6000 0.3654 0.1344 6.4176 1.3 0.1

#E1   E2   E3   epsil  sigma  a_ij a_ik lambda gamm_ij gamm_ik costheta0       Aij
Bij          pij    qij    cij  tol

 Si(a) Si(a) Si(a) 2.1696 2.0951 1.80 1.80 21.0   1.20    1.20    -0.333333333333
7.049556277   0.6022245584 4.0     0.0     1.80 0.0

 Si(a) Si(a) O    2.1696 2.0951 1.90 1.40 10.667 1.93973 0.25    -0.333333333333
7.049556277   0.6022245584 4.0     0.0     1.80 0.0

 Si(a) O    Si(a) 2.1696 2.0951 1.90 1.40 10.667 1.93973 0.25    -0.333333333333
115.364065913 0.9094442793 2.58759 2.39370 1.40 0.0

 Si(a) O    O    2.1696 2.0951 1.65 1.65 3.1892 0.3220  0.3220  -0.333333333333
115.364065913 0.9094442793 2.58759 2.39370 1.40 0.0

 O    Si(a) Si(a) 2.1696 2.0951 1.40 1.40 2.9572 0.71773 0.71773 -0.6155238
115.364065913 0.9094442793 2.58759 2.39370 1.40 0.0

 O    Si(a) O    2.1696 2.0951 0.0  0.0  0.0    0.0     0.0     0.0            115.364065913
0.9094442793 2.58759 2.39370 1.40 0.0

 O    O    Si(a) 2.1696 2.0951 0.0  0.0  0.0    0.0     0.0     0.0            -12.292427744 0.0
0.0    2.24432 1.25 0.0
```

```
#O     O     O     2.1696 2.0951 0.0  0.0  0.0     0.0     0.0     0.0          -12.292427744 0.0
0.0     2.24432 1.25 0.0

#Parameters for the bond-softening function
#Soft E1     E2     m1      m2      m3      m4      m5      R   D
 Soft O     Si(a) 0.0970 1.6000 0.3654 0.1344 6.4176 1.3 0.1
```