AN ABSTRACT OF THE THESIS OF

David Wolff for the degree of Doctor of Philosophy in Physics presented on May 5, 1999. Title: Molecular Dynamics Applications and Techniques: A Comparison Study of Silica Potentials and Techniques for Accelerating Computation

Redacted for Privacy

Abstract approved: _____

Walter Rudd

This thesis presents a study of applications and techniques for molecular dynamics simulations. Three studies are presented that are intended to improve our ability to simulate larger systems more realistically.

A comparison study of two and three-body potential models for liquid and amorphous SiO_2 is presented. The structural, vibrational, and dynamic properties of the substance are compared using two- and three-body potential energy models against experimental results. The three-body interaction does poorly at reproducing the experimental phonon density of states, but better at reproducing the Si-O-Si bond angle distribution. The three-body interaction also produces much higher diffusivities than the two-body interactions.

A study of tabulated functions in molecular dynamics is presented. Results show that the use of tabulated functions as a method for accelerating the force and potential energy calculation can be advantageous for interactions above a certain complexity level. The decrease in precision due to the use of tabulated functions is negligible when the tables are sufficiently large. Finally, an investigation into the benefits of multi-threaded programming for molecular dynamics is presented.

Molecular Dynamics Applications and Techniques: A Comparison Study of Silica Potentials and Techniques for Accelerating Computation

by

David Wolff

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Completed May 5, 1999
Commencement June 1999

Doctor of Philosophy thesis of David Wolff presented on May 5, 1999

APPROVED:

Redacted for Privacy

Major Professor, representing Physics

Redacted for Privacy

Chair of the Department of Physics

Redacted for Privacy

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

David Wolff, Author

ACKNOWLEDGMENT

Thanks to Dr. Walter Rudd for assistance, support, thoughtful discussions, encouragement, and patience throughout this long project. I would also like to acknowledge Dr. Angell of Arizona State University for his advice and the sharing of his considerable expertise in the field of materials science. Thanks also to Dr. Henri Jansen for his advice and much needed encouragement.

CONTRIBUTION OF AUTHORS

Dr. Walter Rudd was involved in the writing and data-collelction for the tabulated potentials study. He also assisted in the interpretation of the data for all of the studies.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure | Page

LIST OF TABLES

DEDICATION

I dedicate this thesis to my family. To my parents, Tony and Kathy, without their love, encouragement, patient understanding, and support I would certainly not have finished this degree. I also dedicate this thesis to my brother Matthew and my sister Allison.

MOLECULAR DYNAMICS APPLICATIONS AND TECHNIQUES: A COMPARISON STUDY OF SILICA POTENTIALS AND TECHNIQUES FOR ACCELERATING COMPUTATION

1. INTRODUCTION

Computational science is an emerging field wherein scientists seek to gain understanding of the real world through the use of high performance computers. The speed at which computers can do mathematical calculations enables researchers to attack problems which were formerly intractable by traditional methods. The use of high speed computers also allows us to simulate the time evolution of highly complex physical systems. Examples include climate modeling, traffic simulations, and the simulation of atomic and molecular systems.

There are numerous so-called grand-challenge problems for which it is hoped that computational science will achieve breakthroughs. These include global climate modeling, electronic structure of materials, pollution and dispersion, genome sequencing, and many more. It is hoped that the use of high performance computers will enable us to gain a much deeper understanding of these important problems.

Of particular interest to this thesis is the grand challenge problem of material properties. The "holy grail" is to reach a point where we are able to simulate a material of macroscopic size (10^{23} atoms) using exact quantum-mechanical many-body calculations. At the present time and in the foreseeable future, this sort of calculation is too complex to be done in a reasonable amount of time (even in a lifetime). Therefore, many approximations are made, such as density functional

theory (DFT), Car-Parinello simulations, Hartree-Fock calculations, and molecular dynamics.

Molecular dynamics is a simulation technique where the basic assumption is that the atoms are described by point particles that interact classically using potential energy functions that are usually developed using approximate quantum-mechanical calculations. The point particles have an effective size which is defined by the properties of the interaction potential energy function.

This is an approximation, of course, since we are dealing with atomic systems and are not doing strict quantum-mechanical calculations. Nevertheless, it has been discovered in recent years that this approximation gives valid results for many of the systems that have been investigated.

In order to move closer to the goal of simulating large systems in detail, we present a study that will help molecular dynamics simulations run faster and better.

Of critical importance to the validity of molecular dynamics simulations is the choice of the inter-atomic potential energy function, because it defines everything about how the system behaves. In Chapter 3 we present a comparison of three potential energy functions which have been proposed for SiO_2. This study is intended to provide researchers with a single source to help determine which potential energy function most accurately simulates SiO_2 under molecular dynamics.

In Chapters 4 and 5, we present studies of two techniques for accelerating calculations within a molecular dynamics simulation, first, by the use of tabulated potential energy functions, and second, by the use of multi-threaded programming techniques. Again, these tools will enable researchers to simulate larger systems, thereby gaining a deeper understanding into how matter behaves.

2. MOLECULAR DYNAMICS

2.1. Background

Molecular dynamics (MD) is simply the simulation of interacting atomic scale particles using classical dynamics. It is a powerful tool for studying the physical and chemical properties of many solids, liquids, and amorphous materials. Recent strides in computer technology, and research into more efficient algorithms, have led to a revolution in computer simulation and in MD.

The conceptual foundation of molecular dynamics is fairly simple, but the method is useful and powerful. For short- to mid-range phenomena, it models reality quite well. Several standard texts on the molecular dynamics method [1–3], and free codes [4–7] are available. Some codes, such as SPaSM which was written at Los Alamos National Laboratory [8], are proprietary.

Molecular dynamics has been used to simulate the properties of many different substances including biological molecules, polymers, nano–scale devices, solids, amorphous materials, and many liquids. A particularly fertile and interesting application for MD has been the simulation of cracks in solids [9–11]. A good survey article on the computing aspects of molecular dynamics was published by S. Gupta [12].

The basic premise is simple. Start with a collection of atoms. For each atom in the simulation, calculate the force on it due to every other atom, integrate Newton's equations of motion for a short increment of time, move all the atoms accordingly, and repeat the process for a large number of timesteps. Timesteps are generally on the order of femtoseconds. The time required for a computer to compute a single timestep depends on many factors including the processor speed, the form of the inter-atomic potential, the size of the simulation, the "tuning" of

the algorithm, and perhaps whether or not the simulation can run in parallel. A single timestep in a useful simulation might take as much as 2-5 seconds to complete. Hence, we are limited in the length of time (that is, both the time of the system being simulated and computational time) for which we can simulate a system. Often it is necessary to compute the mechanical and thermal properties of a system after it has been well thermalized, i.e. after the system has had a long enough time to come into equilibrium and lose all "memory" of its initial state. This can take on the order of hundreds of picoseconds of simulation time. Hence, many simulations of equilibrium properties require hundreds of thousands of timesteps. Simulations involving transport and dynamic properties require even more time.

Such simulations are easily done on sequential machines for small simulation sizes (around 10^3–10^4 atoms) and for larger system sizes with fairly simple potentials. However, in order to simulate physical systems such as nano-sized materials and phenomena such as crack propagation and fracture, we need large system sizes – on the order of 10^5 to 10^9 atoms. Clearly, for these types of systems, sequential computers are not effective, so parallel machines must be employed.

Typically, molecular dynamics algorithms are parallelized for MIMD (Multiple Instruction Multiple Data) machines. These are machines for which each processor has its own proprietary memory space, and memory is shared by message passing between processors. There are many highly efficient algorithms for these architectures [13–21]. In addition, there have been some studies involving parallel techniques which relax inter–processor synchronization to provide speed gains [22–25].

There is great interest in improving our ability to do molecular dynamics in both the temporal and spatial dimensions. In addition to parallel algorithms, there has been much interest recently in coupling molecular dynamics with other types of simulation [26]. The so-called multi-scale problem involves coupling MD with

finite element techniques or quantum-mechanical calculations to span the length scales in a seamless and simultaneous manner. Such techniques allow investigators to simulate much larger systems than are allowed by MD by using MD only on the parts of the system that require that level of precision.

Another interesting issue that has only recently been addressed is the so–called steering techniques. Computational steering is used to refer to the ability to stop, start, and modify a simulation while it is in progress. The large amount of data that is produced by MD simulations make it necessary to have this capability to help decide when to take important measurements. The objectives are to save disk space and to recover from problems without starting the entire simulation over again. Clearly, it would also be useful to be able to visualize the simulation in an interactive manner while it is in progress. This is especially clear for crack simulations. The visualization problem fits well within the computational steering framework for MD. David Beazley has done some interesting work in this area using the scripting language Python and some simple visualization tools [27–30].

Clearly, MD is still limited by its computational complexity. In general, it can be said that bigger is better when it comes to MD. Finite size effects [31] can cause problems with simulations involving small system sizes, and the ability to compare MD results with macroscopic phenomena would be immensely valuable to the advancement of the theory of matter. The rapid advancements in processor speeds, cache technology, and memory speeds may eventually solve this problem for us. However, because bigger will always be better, advanced techniques for accelerating the computations will always be valuable.

2.2. The Basic Strategy

In a MD simulation, the following basic steps [1–3] are carried out in each timestep:

1. Accumulate the force and potential energy (which is usually defined as a function of the position of the atom relative to the other atoms in the system) affecting each atom by calculating the distance to all of the other atoms. For short-range interactions, we can reduce the computational load by considering only neighboring atoms that are within a certain cutoff distance r_c;

2. Integrate Newton's equations of motion for each atom using one of several numerical integration algorithms;

3. Apply boundary conditions;

4. Take measurements (if necessary);

5. Repeat steps 1–4 as often as is necessary to simulate the system on the desired time scale.

Of course there are many variations to this, most of which are methods to reduce the complexity of the force accumulation step, but for the most part the basic outline is the same.

Clearly, the key to having a successful simulation of a particular substance is the choice of the inter-atomic potential energy function. Often the function is just simply a pairwise one, but in order to simulate directional bonding effects, three- and even four–body interactions are becoming more common [32, 33].

2.3. Statistical Mechanics

Molecular dynamics is a method for computing the phase-space trajectory of a system of N particles. In most cases the phase space is a $6N$-dimensional hyperspace consisting of the positions and momenta of the N particles. An MD simulation samples trajectories of the system through this phase space as a function of time. The classical ergodic hypothesis states that the representative point of a single isolated system spends equal amounts of time, over a long period of time, in equal volumes of phase space between the surfaces $E =$ constant and $E + \delta E =$ constant, where δE is arbitrarily small [34]. Hence, MD results over time approximate the ensemble averages of various quantities by using time averages. A conventional MD simulation samples phase–space based on the microcanonical ensemble (NVE) in which the system is considered to be isolated from its surroundings. In this ensemble the number of atoms and the volume or density are kept constant. Energy is automatically conserved. Another common ensemble used in MD is the canonical ensemble (NVT) in which the number of atoms, the volume, and the temperature are held constant. The temperature is usually regulated using one of several techniques [1]. For example the simulation could be coupled to a external heat reservoir by means of auxiliary variables that are introduced into the Lagrangian. Other ensembles that are sometimes used include the isothermal–isobaric ensemble (NPT) in which both temperature and pressure are regulated in addition to the number of particles. Pressure is regulated by varying the size of the simulation region [1].

For a system of N particles, the Lagrangian equation of motion applies:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_k}\right) - \left(\frac{\partial \mathcal{L}}{\partial q_k}\right) = 0 \,, \tag{2.1}$$

where $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$ is the Lagrangian and for point particles, the generalized coordinate vector $\mathbf{q}$ is a $3N$-dimensional vector containing the positions of all the atoms,

$$\mathbf{q} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, ..., \mathbf{r}_N) \ . \tag{2.2}$$

The Lagrangian is defined in terms of the kinetic ($\mathcal{K}$) and potential ($\mathcal{V}$) energies,

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{K}(\dot{\mathbf{q}}) - \mathcal{V}(\mathbf{q}) \ , \tag{2.3}$$

where

$$\mathcal{K}(\dot{\mathbf{q}}) = \sum_{i=1}^{N} \frac{m_i}{2} \dot{r}_i^2 \ . \tag{2.4}$$

To reduce the complexity of the calculations, we expand the potential energy

$$\mathcal{V}(\mathbf{q}) = \sum_{i=1}^{N} \phi_1(\mathbf{r}_i) + \sum_{j>i}^{N} \phi_2(\mathbf{r}_i, \mathbf{r}_j) + \sum_{k>j>i}^{N} \phi_3(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + ... \ , \tag{2.5}$$

where ϕ_k describes the k-body interactions. We usually truncate this series to a small number of terms, often only including the pairwise term. Combining equations 2.3–2.5 with equation 2.1 we get the standard set of Newton's equations,

$$m_i \ddot{\mathbf{r}}_i = -\nabla_{\mathbf{r}_i} \mathcal{V} \qquad i = 1, ..., N \ . \tag{2.6}$$

When the simulation is being done in the NVE ensemble, the total energy, or the Hamiltonian is constant,

$$\mathcal{H}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{K}(\dot{\mathbf{q}}) + \mathcal{V}(\mathbf{q}) = E \ . \tag{2.7}$$

For other ensembles, the equations of motion may be modified. For example to simulate a system in the NVT ensemble we introduce the effects of an external heat bath by the insertion of auxiliary variables into the Lagrangian and hence lose the restriction that total energy remain constant.

2.4. Simulation Methods

In this section I describe briefly some of the tricks and techniques used in a typical MD simulation.

2.4.1. Integration methods

The problem of integrating the equations of motion is: given the position, velocity, and accelerations at time t, to determine the position and velocity at a later time $t+\delta t$ to within a certain degree of accuracy. The choice of the time interval δt depends upon the integration method used, ordinarily δt is much smaller than the time it takes an atom to travel one molecular diameter.

Assuming that the trajectories are continuous, we can expand the position and velocity in a Taylor expansion about time t,

$$\mathbf{r}^p(t+\delta t) = \mathbf{r}(t) + \delta t\,\mathbf{v}(t) + \frac{1}{2}\delta t^2\,\mathbf{a}(t) + \frac{1}{6}\delta t^3\,\mathbf{b}(t) + \cdots \tag{2.8}$$

$$\mathbf{v}^p(t+\delta t) = \mathbf{v}(t) + \delta t\,\mathbf{a}(t) + \frac{1}{2}\delta t^2\mathbf{b}(t) + \cdots \tag{2.9}$$

$$\mathbf{a}^p(t+\delta t) = \mathbf{a}(t) + \delta t\,\mathbf{b}(t) + \cdots\,, \tag{2.10}$$

where $\mathbf{b}(t)$ is the first time derivative of the acceleration. The superscript indicates that these are the "predicted" values for the position and velocity. Given these updated positions and velocities, we can calculate the accelerations (forces) at time $t+\delta t$ which can be compared with the predicted values for the accelerations $\mathbf{a}^p(t+\delta t)$ to get a measure of the size of the error in the prediction step:

$$\Delta\mathbf{a}(t+\delta t) = \mathbf{a}^c(t+\delta t) - \mathbf{a}^p(t+\delta t)\,. \tag{2.11}$$

The predicted values for the position and velocity can then be updated using this value,

$$\mathbf{r}^c(t+\delta t) = \mathbf{r}^p(t+\delta t) + c_0 \Delta\mathbf{a}(t+\delta t) \tag{2.12}$$

$$\mathbf{v}^c(t+\delta t) = \mathbf{v}^p(t+\delta t) + c_1 \Delta\mathbf{a}(t+\delta t)\ . \tag{2.13}$$

Gear [35] has determined the best choice for the coefficients $c_0, c_1, \cdots$. This predictor-corrector method of correcting the predicted values can be iterated, and should eventually converge to the exact values. However, the force calculation part of MD is the most expensive operation and hence one would like to limit the number of times the corrector step is iterated.

There are several predictor-corrector algorithms commonly in use [1], some of which require the positions, velocities, and accelerations to be stored for up to two timesteps. A very good algorithm that does not have this requirement is the so called " velocity Verlet" method. This method takes the form,

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \delta t\, \mathbf{v}(t) + \frac{1}{2}\delta t^2\, \mathbf{a}(t) \tag{2.14}$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \frac{1}{2}\delta t\ [\mathbf{a}(t) + \mathbf{a}(t+\delta t)]\ . \tag{2.15}$$

In this algorithm the force calculation is "sandwiched" in between the prediction and correction steps. Prior to the force calculation, the positions at time $t+\delta t$ and the velocities at mid timestep are calculated using Equation 2.14 and

$$\mathbf{v}(t+\frac{1}{2}\,\delta t\,) = \mathbf{v}(t) + \frac{1}{2}\,\delta t\,\mathbf{a}(t)\ , \tag{2.16}$$

respectively. The forces are then evaluated and the move is completed by calculating the velocities at $t+\delta t$,

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t+\frac{1}{2}\,\delta t\,) + \frac{1}{2}\delta t\,\mathbf{a}(t+\delta t)\ . \tag{2.17}$$

This is not strictly a predictor-corrector algorithm, but it has similar qualities. Its simplicity and numerical stability make it a very effective and economical choice; it is the method used for the simulations performed in this thesis.

Since the velocity Verlet method relies on Taylor expansions of the position and velocity, the error is proportional to a power of the timestep δt. Allen and Tildesley [1, p. 78] state that for Verlet methods, the position has an error of order δt^4 while the velocity has an error of order δt^2. No integration method will give an essentially exact solution to Newton's equations for a very long time. However, exact classical trajectories are not necessary. It is more important that the trajectories be correct for at least as long as the correlation times of interest. See Allen and Tildesley for a discussion of divergence of trajectories [1, p. 76].

2.4.2. Potential Energy Functions

As mentioned in Section 2.3, potential energy functions are often defined in terms of the pairwise and three-body interactions as seen in Equation 2.5. In most MD simulations, there is no external field so the ϕ_1 term is not included. There is almost always a two–body component to the inter-particle interaction. For ionic systems the two-body term includes a coulombic component.

To determine the total force on each atom and the total potential energy, one needs to calculate the force and potential energy between each atom and every other atom in the simulation. For two-body interactions, his requires a total of N^2 calculations. We can reduce the computational effort by taking into account the fact that $\phi(r_{ij}) = \phi(r_{ji})$ and $\mathbf{f}_{ij} = -\mathbf{f}_{ji}$. Doing so reduces the number of calculations by a factor of two, but it still scales as $O(N^2)$. In general, including k-body interactions requires $O(N^k)$ calculations for the accumulation of potential energy and forces in each timestep.

Most potential energy functions, excluding the electrostatic interaction, are short ranged. That is, they decay to zero quickly enough such that the interaction

can be neglected outside of some distance which is shorter than half the size of the simulation box. For functions of this type, we define a cutoff length r_c so that only atoms that are within a distance r_c of each other are considered in the force calculation. For simulations of materials that have a constant density, this reduces the number of atoms that each atom interacts with to a constant number, since each atom now only interacts with other atoms within a sphere of radius r_c. For systems in which this approximation is reasonable, using a cutoff radius reduces the problem from complexity $O(N^k)$ to a problem that scales as $O(N)$.

The electrostatic interaction, unfortunately, does not decay to zero quickly enough allow a cutoff. Some researchers [36] will introduce an exponentially decaying term to the electrostatic interaction in order to simulate screening effects of the surrounding charge distribution, thus allowing the use of a cutoff. However, there are techniques for calculating the full electrostatic interaction with better than $O(N^2)$ scaling. The popular Ewald sum [37] method is discussed later in Section 2.4.6. This method allows the calculation of the electrostatic interaction with periodic boundary conditions in $O(N^{3/2})$ time. There are also the family of so-called fast multipole algorithms [38–41] that can calculate the full electrostatic interaction in $O(N)$ time, and they can be extended to include the effects of periodic boundary conditions.

Three–body interactions are somewhat more complicated to calculate. They involve a triple sum over all atoms within a cutoff length and hence require many more calculations. This can scale as $O(N^3)$ if all interactions are considered. However, Nakano *et al.* have developed a way of decomposing this interaction for the case of AX_2 type systems (SiO_2 , $SiSe_2$, *etc.*) so that the three–body potential energy can be separated into a two–body form [16].

2.4.3. Periodic boundary conditions

The choice of boundary conditions depends greatly on the type of material being simulated. Often when one is simulating a solid, surfaces are important and must be considered. However, when the bulk properties of a liquid or a solid are of interest it is often beneficial to use periodic (or sometimes called toroidal) boundary conditions (PBCs).

PBCs can be thought of as surrounding the MD box with an infinite number of copies of itself. Or another way of thinking of it is that the simulation box is a three-dimensional torus. Each side of the box is associated with the opposite side, so that an atom that moves past one boundary instantly emerges from the opposite boundary. In this space, we can simulate an infinite solid or liquid. Of course, this is not quite accurate since there is periodic symmetry. This effect hides long-wavelength fluctuations. For example, for a box of side length L, the periodicity suppresses density waves with a wavelength greater than L.

One must be careful when simulating a system that uses periodic boundary conditions that the interaction is short-ranged enough that a particular atom does not "feel" the periodicity of the system.

2.4.4. Link–cell method

One problem with force calculations is determining which atoms are close enough with each other to interact. A typical method for solving this problem is the link–cell method. The atoms are sorted into cells which are arranged in a regular lattice throughout the simulation box. Each cell is represented internally by a linked list of atoms. Each atom within a given cell then interacts only with atoms in nearby

cells that are within r_c of the cell. Every few timesteps, the atoms are resorted into cells to account for atoms that have moved outside of their own cell.

This method reduces the amount of work that the computer needs to do to determine where a particular atom's neighbors are located. If the atoms were kept in an array for example, the entire array would have to be searched each timestep to determine which atoms were within r_c of the given atom.

2.4.5. Neighborlists

Another method that further improves performance is the periodic computation of neighborlists. A neighborlist is a list for each atom of the atoms that are within r_c of itself. When the force computation step is done, the potential energy and force are computed by traversing through each atom's neighborlist and calculating the interactions due to those atoms.

The disadvantage to neighborlists is the large amount of storage space required. In many cases a combination of the link–cell method and the neighborlists method is used. The atoms are first sorted into cells to make the calculation of the neighborlists quicker.

2.4.6. The Ewald sum

Due to its long range, the electrostatic interaction is rather difficult to incorporate into MD without losing $O(N)$ scaling. Since the Coulomb interaction extends beyond the size of most simulation regions, the interactions between all pairs must be accumulated. In addition, the interactions between atoms and their periodic images must also be computed! Hence, the scaling for the electrostatic interaction can be worse than $O(N^2)$. In this study I use the Ewald sum because of its simplicity

$\mathbf{n}$	Lattice vector of the periodic array of MD cells.
$\mathbf{k}$	Reciprocal vector of the periodic array of MD cells: $\mathbf{k} = 2\pi\mathbf{n}/L^2$.
L	Size of the edge of the simulation box.
N	Number of atoms.
q	Charge in coulombs.
α	real/reciprocal space partition parameter.
V	volume of MD cell.
$\mathrm{erfc}(x)$	The complimentary error function: $\mathrm{erfc}(x) = \frac{2}{\pi^{1/2}} \int_x^\infty \exp(-t^2)dt$.

TABLE 2.1. Ewald sum parameters.

and because of its ability to handle PBCs. It is also relatively efficient as opposed to fast multipole methods for small system sizes.

To use the Ewald sum, we divide the Coulomb interaction into two sums, one in real space and one in Fourier (reciprocal) space,

$$\begin{aligned}\phi = & \frac{1}{4\pi\epsilon_0} \sum_{\mathbf{n}}^{\dagger} \sum_{j>i} q_i q_j \frac{\mathrm{erfc}\left(\alpha \left|\mathbf{r}_{ij} + \mathbf{n}\right|\right)}{\left|\mathbf{r}_{ij} + \mathbf{n}\right|} \\ & + \frac{1}{\epsilon_0 V} \sum_{\mathbf{k}>0} \frac{1}{k^2} e^{-\frac{k^2}{4\alpha^2}} \left\{ \left| \sum_{i=1}^{N} q_i \cos\left(\mathbf{k}\cdot\mathbf{r}_i\right) \right|^2 + \left| \sum_{i=1}^{N} q_i \sin\left(\mathbf{k}\cdot\mathbf{r}_i\right) \right|^2 \right\} \\ & - \frac{\alpha}{4\pi^{\frac{3}{2}}\epsilon_0} \sum_{i=1}^{N} q_i^2 \,. \end{aligned} \tag{2.18}$$

With appropriate choice of parameters, and the addition of the fast fourier transform method (FFT), this computation can scale as $O(N^{3/2})$. The "daggered" sum indicates the omission of the pair $i = j$ only when $\mathbf{n} = (0, 0, 0)$. An explanation of the other terms in the interaction are shown in Table 2.1.

The force is, of course, just the negative of the gradient of the potential energy:

$$\begin{aligned}\mathbf{f}_i &= -\nabla_{\mathbf{r}_i}\phi \\ &= \frac{q_i}{4\pi\epsilon_0}\sum_{\mathbf{n}}{}^{\dagger}\sum_{\substack{j=1\\ j\neq i}}^{N} q_j \left\{ \frac{\operatorname{erfc}\left(\alpha\left|\mathbf{r}_{ij}+\mathbf{n}\right|\right)}{\left|\mathbf{r}_{ij}+\mathbf{n}\right|} + \frac{2\alpha}{\sqrt{\pi}} e^{-\alpha^2\left|\mathbf{r}_{ij}+\mathbf{n}\right|^2} \right\} \frac{\mathbf{r}_{ij}+\mathbf{n}}{\left|\mathbf{r}_{ij}+\mathbf{n}\right|^2} \\ &+ \frac{2}{\epsilon_0 V}\sum_{k>0} q_i \frac{\mathbf{k}}{k^2} e^{-\frac{k^2}{4\alpha^2}} \\ &\quad \left\{ \sin\left(\mathbf{k}\cdot\mathbf{r}_i\right)\sum_{j=1}^{N} q_j \cos\left(\mathbf{k}\cdot\mathbf{r}_j\right) - \cos\left(\mathbf{k}\cdot\mathbf{r}_i\right)\sum_{j=1}^{N} q_j \sin\left(\mathbf{k}\cdot\mathbf{r}_j\right) \right\} . \end{aligned} \tag{2.19}$$

The parameter α determines when each sum is terminated. Large values for α put more weight on the real space sum and less on the reciprocal space sum, and *vice versa.* If α is chosen small enough, then only terms with $\mathbf{n} = \mathbf{0}$ are included in the real space sum, and the real space part reduces to the normal minimum image convention. We define cut-off values r_c and k_c for which only terms with $|\mathbf{r}_{ij}+\mathbf{n}| < r_c$ and $|\mathbf{k}| < k_c$ are included in the sum. To achieve an $O(N^{3/2})$ scaling, one can use the following choice for α [42],

$$\alpha = \sqrt{\pi}\left(\frac{t_R}{t_F}\frac{N}{V^2}\right)^{\frac{1}{6}} , \tag{2.20}$$

where t_R and t_F are the execution times for one term of the real and reciprocal space interactions respectively. A representative value of 5.5 has been used for t_R/t_F for this study. This number is taken from the number used in the MD package "Moldy." [4]. The particular value is not vitally important since it enters the equation as a sixth root.

If a relative accuracy of $\epsilon = \exp(-p)$ is required, then the cut-off distances are

$$r_c = \frac{\sqrt{p}}{\alpha} \tag{2.21}$$

and

$$k_c = 2\alpha\sqrt{p} \ . \tag{2.22}$$

The use of these cutoff lengths as a function of α (which in turn depends on the system size) assures $O(N^{3/2})$ scaling by varying the cutoff with respect to the system size. See Fincham [42] for a more detailed explanation.

2.5. Measurements

Because the temporal evolution of atomic positions, velocities, and forces are available to us in MD, we can directly measure many physical properties. In the following subsections we introduce MD methods for measuring structural and dynamic properties of a MD simulation.

2.5.1. Energy, Temperature, and Pressure

Perhaps the most obvious quantities that one might want to measure are total energy, temperature, and pressure. The instantaneous total energy is of course the sum of the total potential and kinetic energy at a given instant. The potential energy is usually accumulated during the force computation. Once the forces are integrated and we have access to the updated velocities, the total kinetic energy can be calculated.

The temperature is derived from the mean kinetic energy, via the equipartition theorem,

$$T = \frac{2}{3k_B} < \mathcal{K} >= \frac{1}{3Nk_B} \sum_{i=1}^{N} m_i v_i^2 \ , \tag{2.23}$$

where k_B is Boltzmann's constant.

Pressure is calculated from the virial theorem

$$PV = Nk_BT + \frac{1}{3}\left\langle \sum_{i=1}^{N} \mathbf{r}_i \cdot \mathbf{F}_i \right\rangle \ . \tag{2.24}$$

It is often more convenient to use the following form which is independent of the origin of coordinates since the interaction is typically pairwise,

$$PV = Nk_BT + \frac{1}{3}\left\langle \sum_{j>i}^{N} \mathbf{r}_{ij} \cdot \mathbf{F}_{ij} \right\rangle \ . \tag{2.25}$$

Of course, in order to get statistically valid results for these or other quantities, one needs to allow the system to equilibrate, and then average over many timesteps.

2.5.2. Pair Distribution Function

One of the most useful of the structural correlation functions is the pair distribution function. This function is a representation of the average fluctuations in density around any given atom in the simulation. Let $N = N_\alpha + N_\beta$ for a binary system of N particles where N_α and N_β are the number of particles of species α and β. The pair distribution function $g(r)$ is then defined as,

$$\langle n_{\alpha\beta}(r) \rangle = 4\pi r^2 \, \Delta r \, \rho \, c_\beta \, g_{\alpha\beta}(r) \ . \tag{2.26}$$

where $n_{\alpha\beta}$ is the number of particles of species β in the spherical shell between r and $r + \Delta r$ around an individual particle of species α. The angular brackets (<>) represent the ensemble average and an average over all the particles of species α. $c_\beta = N_\beta/N$ is the concentration of species β, and $\rho = N/V$ is the number density.

The total pair distribution function is then defined as

$$g(r) = \sum_{\alpha,\beta} c_\alpha \, c_\beta \, g_{\alpha\beta}(r) \ . \tag{2.27}$$

2.5.3. Structure factors

The static structure factor is an important function because it can be related to the results of neutron scattering experiments. We will revisit the calculation of the static structure factor in our discussion of SiO_2 potential energy functions in Section 3.4.4.

2.5.4. Bond angle distributions

Bond angle distributions are computed by accumulating a histogram of the angles between atoms of various species. For example, in an AX_2 type system (such as SiO_2), the procedure would be as follows. First, for each atom of type A, a list of neighbors would be constructed of type A and of type X. Only neighbors within a certain distance would be added to the neighborlist. This distance is determined by the minimum after the first peak in the partial pair distribution function. The reason for this is because we only want to include nearest neighbors, and the first peak of the pair distribution function describes the position of an atom's nearest neighbor. Once we have these neighborlists, we can compute the angles A-A-X, A-A-A, and X-A-X, being careful not to double count. A similar procedure is done for each X atom to get X-X-A, X-X-X, and A-X-A. Each value for the angle is binned and a histogram is created for each bond angle.

2.5.5. Phonon density of states

The phonon density of states describes the vibrational activity of the system. It indicates the density of normal modes as a function of energy. The partial phonon

density of states can be derived from the Fourier transform of the velocity auto–correlation function [43],

$$F_\alpha(\omega) = \int_0^\tau Z_\alpha(t) \cos(\omega t) e^{-\gamma(t/\tau)^2} dt \ , \tag{2.28}$$

where a Gaussian window function with $\gamma = 1$ and $\tau = 3$ ps is used, and Z_α is the velocity auto–correlation function for the αth species which is given by,

$$Z_\alpha(t) = \frac{\left\langle \sum_{i=1}^{N_\alpha} \mathbf{v}_i(0) \cdot \mathbf{v}_i(t) \right\rangle}{\left\langle \sum_{i=1}^{N_\alpha} v_i^2(0) \right\rangle} \ . \tag{2.29}$$

The total phonon density of states is obtained by summing over the partial density of states with the concentration

$$F(\omega) = \sum_\alpha c_\alpha F_\alpha(\omega) \ . \tag{2.30}$$

2.5.6. Other correlation functions

These are functions that evolve in the time domain and are useful for determining dynamic properties of the system. The averaging technique for these functions is complicated, so I refer the reader to Rapaport for more information on how to correctly average these functions in a simulation [2, p.119].

The diffusion coefficient describes the mean-square displacement of a particle of type α per unit time, and is given by

$$D_\alpha = \lim_{t\to\infty} \frac{\langle r^2(t) \rangle_\alpha}{6t} \ , \tag{2.31}$$

where,

$$\langle r^2(t) \rangle_\alpha = \left\langle \frac{1}{N_\alpha} \sum_{j(\alpha)} [\mathbf{r}_j(t+s) - \mathbf{r}_j(s)]^2 \right\rangle \ , \tag{2.32}$$

is the mean-square displacement.

This quantity can be compared with experimental values to determine the validity of the potential energy function. It can also be used to determine the temperature of transitions from the liquid to the solid or amorphous state (see Section 3.4.3).

The velocity auto-correlation function conceptually describes variations in an average particle's direction of travel, and can be used to calculate the phonon density of states. It can also be used as an alternative method for calculating the diffusion coefficient. Equation 2.28 gives its definition.

3. A MOLECULAR DYNAMICS STUDY OF TWO AND THREE-BODY POTENTIAL MODELS FOR LIQUID AND AMORPHOUS SILICA

David A. Wolff and Walter G. Rudd

To be submitted to *Physical Review B*

3.1. Introduction

Molecular dynamics is an effective tool for the study of supercooled liquids and glasses [44–46]. Computer simulation is particularly suited for materials of this nature for several reasons. First, computer simulation allows the investigation of the material in microscopic detail, including the full trajectory of each atom. Second, it allows a view of the evolution of the system on a time scale much smaller than can typically be seen in experiment. Since supercooled liquids and glasses have a non–periodic structure, these properties are particularly valuable.

Amorphous silica (SiO_2) has historical, technological, and scientific importance and is perhaps one of the most extensively studied materials in condensed matter science. Because of the absence of a periodic structure, it is difficult to determine its long-range, three-dimensional properties. One of the most successful models for the structure of SiO_2 glass is the continuous random network model [47]. In this model the glass is formed by a random network of corner-sharing SiO_4 tetrahedra with a wide distribution of Si–O–Si bond angles centered at approximately 142°.

The structure of SiO_2 glass has been extensively studied by neutron scattering experiments [48–52]. The wealth of experimental results has facilitated a great deal of work in computer simulation in an attempt to better understand the microscopic structure of silica. Of particular interest in recent years from both an experimental and simulation standpoint has been the intermediate range order, which is related to the first sharp diffraction peak (FSDP) [53, 31, 54]. Recent work has also focused on the vibrational excitations and frequency spectrum of amorphous silica [55–58], including the infrared spectrum [59–61].

Classical molecular dynamics has its limitations. Ultimately, the more accurate "Car–Parinello" approach [62] in which the forces are determined by high-order quantum-mechanical calculations is necessary to accurately simulate silica systems. Several successful investigations have been carried out using this approach [63–66]. Unfortunately, the utility of this method is limited by the large amount of computer time required to conduct such simulations. For the time being, molecular dynamics simulations have proven quite effective despite their purely classical approach.

In fact, given current computational equipment, classical molecular dynamics simulation is mandatory for many problems, particularly when large systems or long time scales are involved. There have been several such large–scale studies on silica [67–71].

Choosing an inter-atomic potential energy function is a critical step towards an accurate MD simulation of liquid and amorphous silica. There is a real need to evaluate the possible interaction functions proposed for SiO_2 and decide how best to apply the considerable resources and computer time to investigate the properties of interest. Many different potential energy functions have been proposed, but only a few have produced qualitatively accurate results across a broad range of properties. In this study we compare the results obtained from a simulation of silica in the liquid and amorphous state using three of the most popular interactions. We emphasize the comparison between the two–body and three–body models. We build upon a recent study by Hemmati and Angell in which the thermodynamic, angular, and diffusional properties of several pair potential energy models for liquid SiO_2 were compared [72]. The same group, along with Wilson and Madden, have also published some simulation studies of the infrared spectrum of SiO_2 using the same set of functions [59, 60]. In these studies, Angell *et al.* documented the failure of pair interactions to reproduce the separation of the principal peaks of the infrared

spectrum, and their failure of to reproduce the temperature at which the density is a maximum. The density maximum is a liquid state anomaly which is related to the anomalous pressure coefficient of the fluidity [73].

In this study we provide a comparison of the structural, diffusive, and vibrational properties of silica derived using the pair potential energy functions of van Beest, Kramer, and van Santen (BKS) [74] and Tsuneyuki, Tsukada, Aoki, and Matsui (TTAM) [75], versus the same properties derived using the three–body potential energy of Vashista, Kalia, Rino, and Ebbsjö (VKRE) [76].

The structure of this paper is as follows: A brief description of each interaction is given in Section 3.2. The computational procedure for this study is discussed in Section 3.3. Results are discussed in Section 3.4, followed by a discussion and conclusions in Section 3.5.

3.2. Potential Energy Functions

3.2.1. Pairwise Functions

The BKS and TTAM inter-atomic potential energy functions can be written in an exp-6 general form,

$$\phi(r_{ij}) = \frac{q_i q_j}{r_{ij}} + A_{ij} e^{-B_{ij} r_{ij}} - \frac{C_{ij}}{r_{ij}^6} \ , \tag{3.1}$$

where q_i and q_j are the effective charges. We use values of 2.4 e and -1.2 e (where e is the electron charge) for Si and O respectively as suggested by the authors of the TTAM and BKS interactions.

Interactions with this functional form have the unphysical property of diverging to minus infinity at small inter-atomic distances. This can be disastrous at high temperatures, because atoms may have the kinetic energy to overcome the

potential energy barrier and fuse together. Therefore this effect must be avoided by the use of a substitute potential energy function as described in section 3.3.

The TTAM potential energy function was developed for the accurate evaluation of properties in the crystalline state [75]. Derived from *ab initio* Hartree-Fock self-consistent-field calculations, Tsuneyuki *et al.* have demonstrated that four of the known crystalline polymorphs are stable under this interaction. They have also used this function to show the α to β structural phase transition at 850 K [77].

Despite the fact that it was intended to model the solid state of silica, the TTAM interaction is effective in the simulation of the liquid state as well [78]. It also reproduces several properties of the amorphous state as is detailed below.

Hemmati and Angell have shown that the BKS interaction is superior to the TTAM interaction for certain properties of amorphous silica [72, 59]. However, we include the TTAM interaction in our study for comparison purposes.

The BKS interaction is based on *ab initio* calculations and experimental data [74]. The authors claim it is an improvement over the TTAM potential energy function and compare several structural observables between the two.

The BKS interaction has the same functional form as the TTAM interaction. The only important difference between the two is that for the BKS potential energy the Si-Si interaction has no short range term. See Table 3.1 for the values of the constants.

	BKS	TTAM
A_{OO}	1388.7730	1746.70
B_{OO}	2.76000	2.84091
C_{OO}	175.0000	214.91
A_{SiO}	18003.7572	10096.06
B_{SiO}	4.87318	4.784689
C_{SiO}	133.5381	70.81
A_{SiSi}	0.0	5.95184×10^{8}
B_{SiSi}	0.0	15.1515
C_{SiSi}	0.0	0.0

TABLE 3.1. Parameters for the exp-6 silica pair potentials. A is in eV, B is in Å^{-1}, and C is in eV Å^6.

3.2.2. VKRE Potential

The VKRE potential energy [76] contains an additional three–body term in order to simulate bond stretching and bending effects,

$$\Phi = \sum_{i<j}^{N} \phi_{ij}^{(2)} + \sum_{i<j<k}^{N} \phi_{jik}^{(3)} \ . \tag{3.2}$$

The two–body part $\phi_{ij}^{(2)}$ includes three terms, the Coulomb interaction, steric repulsion due to ionic sizes, and a charge–dipole interaction resulting from the large electronic polarizability of O^{2-}. The two body part is

$$\phi_{ij}^{(2)} = \frac{Z_i Z_j}{r_{ij}} + \frac{H_{ij}}{r_{ij}^{n_{ij}}} - \frac{P_{ij}}{r_{ij}^4} \exp\left(\frac{-r_{ij}}{r_{s4}}\right) \ , \tag{3.3}$$

where

$$H_{ij} = A_{ij}(\sigma_i + \sigma_j)^{n_{ij}} \ , \tag{3.4}$$

and

$$P_{ij} = \frac{1}{2}(\alpha_i Z_j^2 + \alpha_j Z_i^2) \ . \tag{3.5}$$

The ionic radii σ_i, repulsive exponents n_{ij}, repulsive strength A_{ij}, effective charges Z_i, and polarizabilities α_i are tabulated in Table 3.2.

The three–body term has the form

$$\phi_{jik}^{(3)} = B_{jik} f(r_{ij}, r_{ik})(\cos\theta_{jik} - \cos\bar{\theta}_{jik})^2 \ , \tag{3.6}$$

where

$$f(r_{ij}, r_{ik}) = \exp\left(\frac{l}{r_{ij} - r_{c3}} + \frac{l}{r_{ik} - r_{c3}}\right) \ , \tag{3.7}$$

and

$$\cos\theta_{jik} = \frac{\mathbf{r}_{ik} \cdot \mathbf{r}_{ij}}{r_{ik} r_{ij}} \ . \tag{3.8}$$

A_{ij} (eV)	r_{s1} (Å)	r_{s4} (Å)	r_c (Å)	l (Å)	r_{c3} (Å)
0.7752	4.43	2.5	5.5	1.0	2.6

	σ_i (Å)	Z_i (e)	α_i (Å^3)
Si	0.47	1.2	0.0
O	1.2	-0.6	2.4

	n_{ij}
Si-Si	11
Si-O	9
O-O	7

	B_{jik} (eV)	$\bar{\theta}_{jik}$
Si-O-Si	19.97	141.0
O-Si-O	5.0	109.47

TABLE 3.2. Parameters for the 3 body interaction for SiO_2 glass from Vashista *et al.*

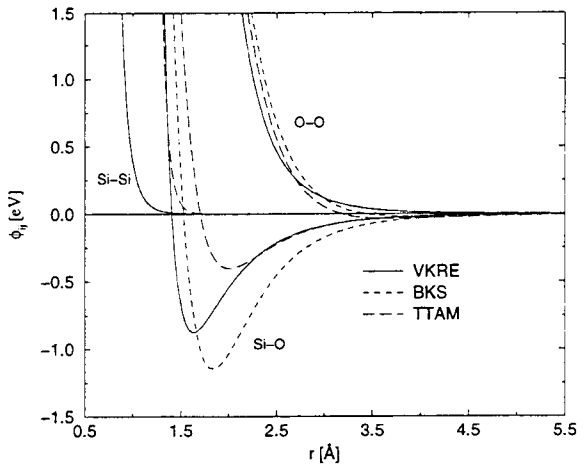

FIGURE 3.1. Potential energy for the three inter-atomic potential energy functions. The two–body part of the VKRE potential energy is shown. The Coulomb interaction is omitted.

The strength of the three–body interaction B_{jik}, average bond angles $\bar{\theta}$, and cut–off distances are tabulated in Table 3.2. The parameters in this study are taken from a more recent publication [36].

A comparison of the two–body parts of all three interactions is shown in Figure 3.1. Figure 3.2 shows the three–body interaction as a function of angle. For the purposes of this figure, the atoms are assumed to form an isosceles triangle where the Si–O distance is given by the average bond length, 1.62 Å.

3.3. Computational Procedure

Our simulation was performed using the linked-list cell method [1]. The equations of motion were integrated using the velocity Verlet integration method [1,

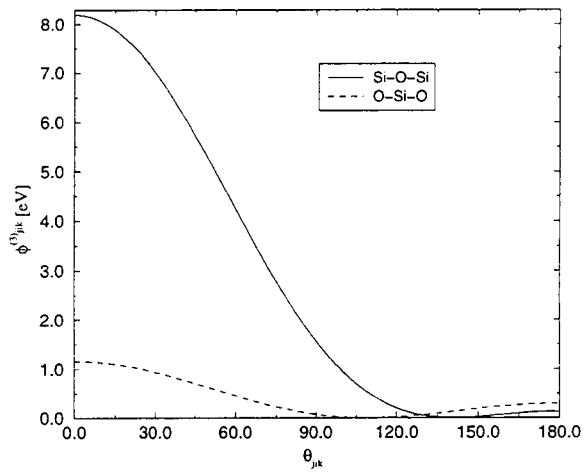

FIGURE 3.2. The three-body energy *vs.* angle for triplets of atoms forming an isosceles triangle. The angle θ_{jik} is between the two bonds that are of equal length (1.62 Å).

p. 81]. Electrostatic interactions were calculated using the well known Ewald sum method [1]. The timestep size was 1 fs.

The system is initialized on an α–quartz lattice at 2.2 g/cm^3 , and is immediately heated to a liquid state at 8,000 K. The temperature is regulated by velocity scaling. For each interaction, three samples are created, each with a different random seed for the calculation of initial velocities. The samples are then cooled according to an interaction-specific cooling scheme. The temperature is regulated in blocks that consist of a scaling phase, an equilibration phase, and a measurement phase. The samples equilibrate between cooling stages in the micro–canonical (NVE) ensemble.

3.3.1. Cooling Scheme for the BKS and TTAM Interactions

For the BKS and TTAM interactions, in the temperature range between 8,000 K and 4,000 K, the temperature is regulated in blocks of 20,000 timesteps (20 ps). In a block, the temperature is reduced for 1 ps, the system equilibrates for 9 ps, and measurements are taken during the final 10 ps. This 20,000 timestep block is repeated to cool the system from 8,000 K to 4,000 K in steps of 1,000 K.

As previously noted, the BKS and TTAM potential energies diverge to minus infinity at small inter-atomic distances. At high temperatures, it is possible that an individual atom may have enough kinetic energy for it to overcome the energy barrier and fuse with another atom. For this reason, the potential energy function is replaced with a simple harmonic potential energy whenever atoms come within 1.2 Å (for Si-O) or 1.7 Å (for O-O) of each other. The Si-Si interaction is not considered here because for both potential energies the coefficient of the diverging term (r^{-6}) is zero.

We use blocks of 30,000 timesteps for cooling the system between 4,000 K and 2,000 K. In each block the system is cooled by 200 K. During the first 10 ps of each block, the system is gradually cooled at a rate corresponding to 0.333×10^{13} K/s. Then during the next 10,000 timesteps the system is allowed to equilibrate in the NVE ensemble. During the final 10,000 timesteps of each block various measurements are taken for that temperature.

After the system has reached 2,000 K, it is again cooled in blocks of 30 ps. However, the system is cooled by simply scaling the velocities directly to the desired temperature during the first 1,000 timesteps of each block. The system is allowed to equilibrate for 19,000 timesteps, and then measurements are taken over

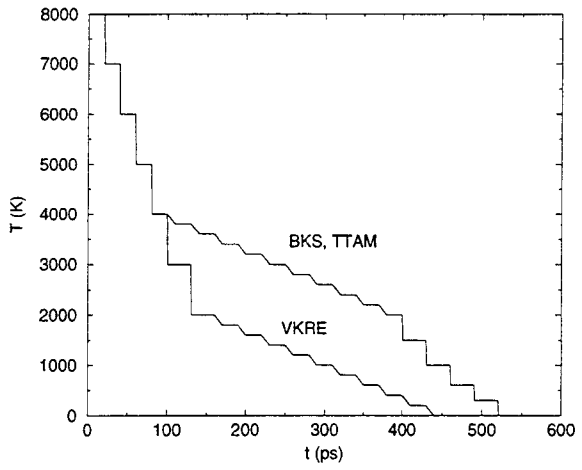

FIGURE 3.3. The cooling scheme for each of the three interactions.

the remaining 10,000 timesteps. This is repeated for the temperatures of 1,500 K, 1,000 K, 600 K, 300 K, and 0 K.

This cooling scheme allows for slow cooling in the range near the amorphous transition temperature. Vollmayr *et al.* [79] have shown that the cooling rate has a direct effect upon the glass transition temperature. They show a logarithmic dependence on the cooling rate. Hence, in this study, I have been careful to use a slow cooling rate in the region near the transition temperature.

3.3.2. Cooling Scheme for the VKRE Interaction

The amorphous transition for the VKRE interaction is at a different temperature from that of the TTAM and BKS interactions. Hence, another cooling scheme is required.

The system is first cooled from 8,000 K to 4,000 K as in the above scheme. For the range 4,000 K to 2,000 K, the system was cooled in blocks of 30,000 timesteps by 1,000 K each, in the same manner as for the range of 2,000 K to 0 K in the previous cooling scheme.

For the range of 2,000 K to 0 K, the system was cooled in blocks of 30,000 timesteps by 200 K each, again using the same method as the previous scheme used for the range 4,000 K to 2,000 K. Figure 3.3 graphically shows the cooling scheme for all three interactions.

3.3.3. Measurements

Measurements for each temperature were evaluated over a period of 10,000 timesteps. Pair distribution functions, bond angle distributions, diffusion constants, and velocity auto–correlation functions were all averaged over all 10,000 steps.

3.4. Results

3.4.1. Pair Distribution Function and Coordination Numbers

We first demonstrate the validity of the simulated system by investigating the pair distribution function and coordination numbers at finite temperatures.

The partial radial distribution functions ($g_{\alpha\beta}(r)$) are calculated from a histogram of pairwise distances,

$$\langle n_{\alpha\beta}(r)\rangle = 4\pi r^2 \Delta r \rho c_\beta g_{\alpha\beta}(r) \ , \tag{3.9}$$

where ρ is the number density, c_β is the concentration of species β (N_β/N), and $n_{\alpha\beta}(r)$ indicates the number of particles of species β in the shell between r and

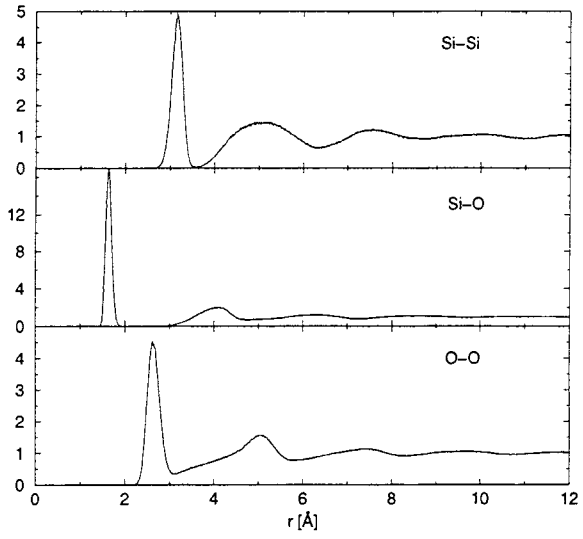

FIGURE 3.4. The partial radial distribution functions $g(r)$ at 1,000 K for the BKS interaction.

$r + \Delta r$ around a particle of species α. The angular brackets represent the ensemble average and an average over particles of species α.

The partial radial distribution functions are shown in Figure 3.4. Because the radial distribution functions for the three interactions are so similar in shape, only the results for the BKS interaction are shown. The temperature is 1,000 K, well below the glass transition temperature.

The positions of the first and second peaks in the radial distribution function are tabulated in Table 3.3. We observe no significant differences in the peak positions and shapes of the radial distribution function between the two-body and three-body interactions. This is expected because the radial distribution has no angular dependence and hence shouldn't be affected by the angular components introduced by the three-body interaction.

		TTAM (Å)	BKS (Å)	VKRE (Å)	Experiment (Å)	
Si-O	first peak	1.63(1)	1.61(1)	1.61(1)	1.608[b]	1.620[a]
	second peak	4.14(2)	4.10(4)	4.18(1)		4.15[a]
O-O	first peak	2.62(1)	2.60(1)	2.65(1)	2.626[b]	2.65[a]
	second peak	5.06(1)	5.01(3)	5.05(5)		4.95[a]
Si-Si	first peak	3.16(1)	3.14(1)	3.13(1)	3.077[c]	3.12[a]
	second peak	5.06(10)	5.12(6)	5.11(6)		5.18[a]

[a]Reference [80].
[b]Reference [52].
[c]Reference [81].

TABLE 3.3. The positions of the peaks in the partial radial distribution functions. The numbers in parenthesis give the estimated error in units of the last digit.

The positions of the first and second peaks for the three interactions agree quite well with experiment; all three potential energy functions seem to reproduce the short- and medium-range structure of the glass.

Coordination numbers ($N_{\alpha\beta}(R)$) indicate the average number of particles of species β around a particle of species α within a sphere of radius R. They are calculated from the integral over the corresponding partial pair distribution function,

$$N_{\alpha\beta}(R) = 4\pi\rho c_\beta \int_0^R g_{\alpha\beta}(r) r^2 dr \ . \tag{3.10}$$

Since we are interested only in nearest neighbors, the value for R is taken to be the first minimum of the corresponding partial pair distribution function. For R, we used values of 2.1, 3.1, and 3.7 Å for the Si-O, O-O, and Si-Si coordination numbers respectively.

	TTAM	BKS	VKRE
Si–O	4.009	3.999	3.968
O–Si	2.005	2.0	1.984
O–O	6.06	6.069	5.999
Si–Si	3.985	4.023	3.968

TABLE 3.4. Coordination numbers.

The coordination numbers are shown in Table 3.4. The results for the three interactions agree with each other very well. The Si–O coordination number agrees with the expected value of 4 which when combined with the bond angle distribution data presented below, is characteristic of the SiO_4 tetrahedral configuration. Again, we observe no significant difference in this quantity between the pairwise and three–body interactions.

3.4.2. Bond Angle Distributions

Using pairwise interactions the Si–O–Si bond angle distribution has been difficult to reproduce accurately through simulation. One of the major reasons for introducing a three–body interaction for silica is to correct this problem.

Amorphous SiO_2 can be represented by a random network of corner-sharing tetrahedra (see Figure 3.5), with a Si-O-Si bond angle of about 142°. The O-Si-O bond angle should be close to the true tetrahedral angle of 109.47°. All three interactions yield this value reasonably well.

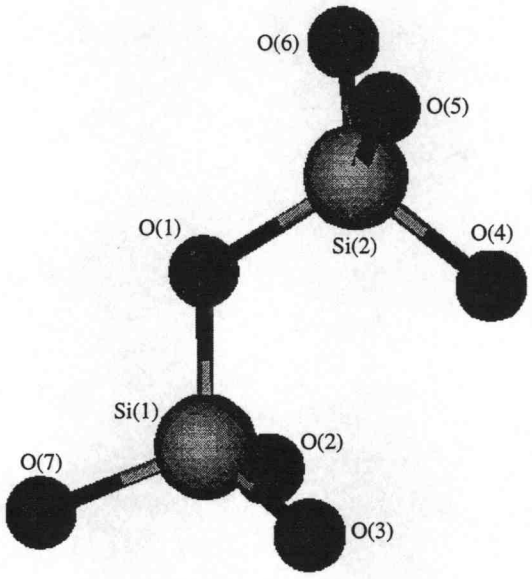

FIGURE 3.5. A graphical representation of corner-sharing tetrahedra.

The Si-O-Si bond angle is created by two neighboring tetrahedra and their shared oxygen atom. The addition of the angular term in the VKRE potential energy has the effect of narrowing the size of the Si-O-Si distribution and changing its position. Figure 3.6a shows the Si-O-Si bond angle distribution for the BKS and VKRE interactions (the TTAM results are similar to the BKS results). The distribution from the BKS interaction is significantly wider and shifted to the right as compared to the distribution from the VKRE interaction. This effect is also seen in Figure 3.6b which shows the first peak of the O-Si-Si distribution. In this case however, the BKS peak is shifted to the left of the VKRE peak. This can be explained when one considers the triangle created by two silicon nearest neighbors and a corner shared oxygen atom. In Figure 3.5 these atoms are O(1), Si(1) and Si(2). Since the VKRE interaction has the effect of restricting the Si-O-Si bond

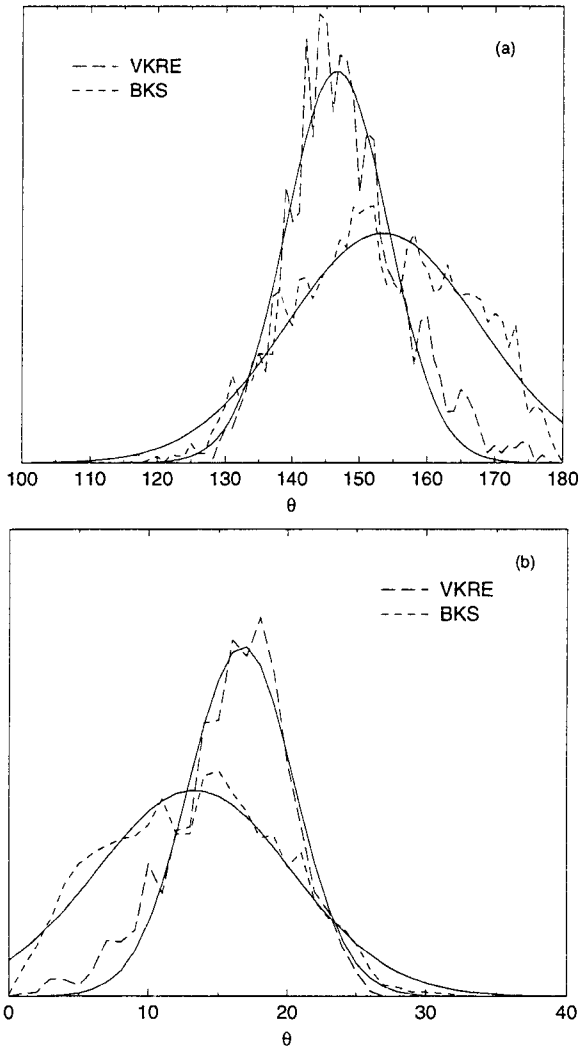

FIGURE 3.6. Bond angle distributions for (a) the Si-O-Si angle and (b) the first peak in the O-Si-Si distribution. The solid lines are Gaussian fits to the data.

	TTAM	BKS	VKRE	Experiment
O–Si–O	108.6° (13.6°)	108.6° (13°)	109° (9°)	109.75°[a]
Si–O–Si	153° (30°)	153.3° (33°)	146° (18°)	144° (38°)[b], 142° (26°)[c], 144°[d], 152°[e]

[a]Computed using the bond lengths reported in Mozzi and Warren [80].
[b]Mozzi and Warren [80]
[c]Pettifer *et al.* [82]
[d]Coombs *et al.* [83]
[e]DaSilva *et al.* [84]

TABLE 3.5. Location and full width at half maximum of the bond angle distributions for the Si-O-Si and O-Si-O angles from each interaction and from experiment.

angle to a smaller angle, it has the opposite effect on the other two angles of the triangle, which are represented in the first peak of the O-Si-Si distribution. The peak position of the VKRE distribution was 16.7° with a width at half maximum of 9°, while the BKS distribution had a peak position of 13.3° and a width at half maximum of 16.7°.

Results for all three interactions are given in Table 3.5. The peak positions and widths at half maximum are evaluated by Gaussian fits to the distribution at 0 K. It is clear from the results that the VKRE results are closer to experimental values for both the Si-O-Si (with the exception of the DaSilva *et al.* result) and the O-Si-O distributions. However, the VKRE interaction underestimates the width at half maximum for the Si-O-Si distribution.

From these results it appears that the VKRE interaction does an overall better job of reproducing the peak positions for the bond angles in silica. However,

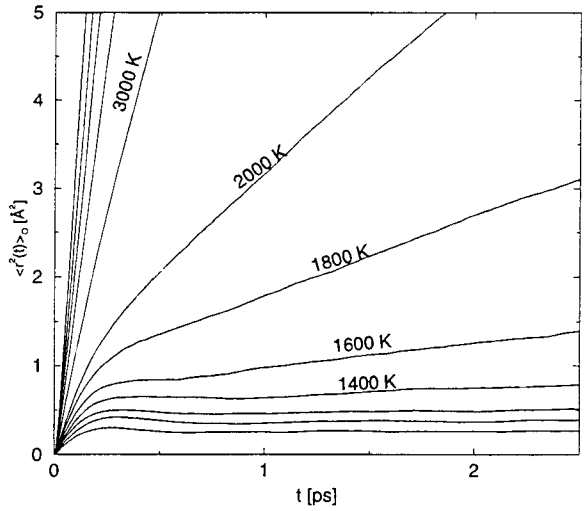

FIGURE 3.7. Mean–squared displacement curves for oxygen for various temperatures for the VKRE interaction.

we found that for all of the other possible angles – O-O-O, Si-Si-Si, and Si-O-O – there is no significant difference between the two-body results and the VKRE results.

3.4.3. Glass Transition and Diffusivities

The diffusion constant for species α can be calculated from the mean–square displacements,

$$D_\alpha = \lim_{t\to\infty} \frac{\langle r^2(t)\rangle_\alpha}{6t} \ , \tag{3.11}$$

where

$$\langle r^2(t)\rangle_\alpha = \left\langle \frac{1}{N_\alpha} \sum_{j(\alpha)} [\mathbf{r}_j(t+s) - \mathbf{r}_j(s)]^2 \right\rangle \ . \tag{3.12}$$

A plot of $\langle r^2(t)\rangle_O$ *versus* time for the VKRE interaction is shown in Figure 3.7. The diffusion constant is derived from the slope of the linear portion of the curve.

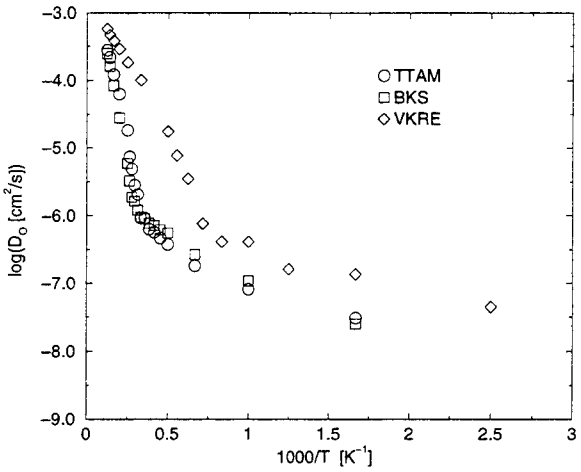

FIGURE 3.8. Arrhenius plot of the diffusion constant of oxygen for the three interactions. The break in the near linear behavior indicates the liquid to amorphous transition.

For low temperatures, the behavior of the mean–square displacement curve is more erratic due to the low rate of diffusion. Hence, it is difficult to get accurate results for the diffusion without increasing the length of the measurement periods.

The "glass transition" as defined in molecular dynamics simulations is not necessarily related to the experimental glass transition due to the vast difference in time scales. The transition to the vitreous or glassy state is essentially defined by the crossing of the liquid relaxation time scale and the external (experimentalist-controlled) observation time scale. The "glass transition" occurs by a breaking of ergodicity when the cooling rate becomes such that the system does not have enough time to become fully equilibrated (ergodic) before the next cooling step. Each successive cooling step then further reduces the degree of equilibration. Because of the fact that molecular dynamics cooling rates are several orders of magnitude faster than what is possible in experiment, and because relaxation times increase

with decreasing temperature, one would expect to see simulated transitions at much higher temperatures than are observed in experiment. Angell and others have done extensive studies on relaxation times and cooling rates and their relation to the glass transition [85–88].

For this study, we attempted to locate the glass transition using the exponential dependence of the diffusion constant on temperature. A plot of the logarithm of the diffusion constant *versus* inverse temperature (Ahrrenius plot) is shown in Figure 3.8. The data shows a nearly linear relationship for the high and low temperature regions for each interaction. We can see that there is a discontinuity in the slope of the curves where the data seems to "flatten out". This flattening out of the Ahrrenius plot indicates the breaking of ergodicity related to the transition to the vitreous or amorphous state. The transition is generally referred to as a range of temperatures rather than a specific temperature; however we define a particular value for the transition for purposes of comparison in the following way. The high temperature data was fit with a straight line and the same was done for the low temperature data. The point of intersection of the two lines is taken as the transition temperature T_g. Our results show a T_g of 2,900 K for the TTAM interaction, 3,300 K for the BKS interaction, and 1,200 K for the VKRE interaction.

Vollmayr *et al.* have results that show the glass transition temperature for the BKS interaction as a function of the cooling rate [79]. They find that T_g varies logarithmically with the cooling rate γ, from 2,900 K for the slowest cooling rate (approx. 10^{12} K/s) to 3,300 K for the highest (approx. 10^{14} K/s). Our results agree with these results for the fastest cooling rates studied despite the fact that our cooling rate corresponds to about the middle of the range of cooling rates in their study. We attribute this difference to the sensitivity of the results to the low temperature data. Since the relaxation times of the system become very long for

low temperatures, the measured diffusivities become less certain. To counter this effect, we have left some of the low temperature data out of the calculations.

It is obvious from Figure 3.8 that the VKRE interaction is much more diffusive than the two-body interactions. This is most likely due to the factor of two difference in the silicon and oxygen effective charges. It is likely that these comparatively low effective charges were chosen such that the simulated system would demonstrate a melting temperature comparable to experimental values. Our results show that the VKRE interaction demonstrates a breaking of ergodicity at a temperature much closer to the experimental value of 1,453 K [89, p.39]. However, as indicated previously, it is not necessary or even desirable that these temperatures match because of the vast difference in time scales.

A more important consideration perhaps is whether or not the diffusion for the system in equilibrium is comparable to experimental results. Hemmatti and Angell have shown that there exists a spread of two orders of magnitude in the diffusivites at 6,000 K for different two-body interactions [72]. Our results show very little spread in the diffusivities for the TTAM and BKS interactions. However, the VKRE interaction's diffusivities differ from the other two by as much as two orders of magnitude at 3,000 K. Hemmatti and Angell also cite experimental diffusivities of around 10^9 cm^2/s at approximately 3,000 K. The VKRE interaction over-estimates this by about five orders of magnitude. Clearly, the VKRE interaction is much too diffusive.

3.4.4. Structure Factors

Since the static structure factor contains no more information than is in the radial distribution function, we do not expect to see any difference between the

results obtained via a three–body interaction. However, the static structure factor is important for comparison with scattering experiments, and can therefore help validate our results.

We compute the structure factor via its formal definition,

$$S(q) = \frac{1}{N} \left\langle \sum_{j,k} e^{i\mathbf{q}\cdot(\mathbf{r}_j - \mathbf{r}_k)} \right\rangle . \tag{3.13}$$

Averaging over all directions for the vector $\mathbf{q}$ the expression becomes

$$S(q) = \frac{1}{N} \left\langle \sum_{j,k} \frac{\sin(qr_{jk})}{qr_{jk}} \right\rangle . \tag{3.14}$$

In order to compare the molecular dynamics result with the results from neutron scattering experiments, $S(q)$ must be weighted with the coherent scattering lengths,

$$S_N(q) = \frac{1}{N\langle b^2 \rangle} \sum_{j,k} b_j b_k \left\langle \frac{\sin(qr_{jk})}{qr_{jk}} \right\rangle , \tag{3.15}$$

where b_j is the coherent neutron scattering length of atomic species j.

Often, the structure factor is computed via the Fourier transform of the radial distribution function. We feel that our method yields more accurate results because all pairs of atoms are included in the average, whereas the pair distribution function includes only information from neighbors within a certain distance.

Our results for the partial structure factors at 0 K are shown in Figure 3.9. As expected, there is little difference in the general shape between the three interactions. The peak positions also match quite well.

The first peak for each of the three partial structure factors is shown in a close up view in the inset graph. This peak is called the first sharp diffraction peak (FSDP) and has been the subject of much scrutiny in recent years [53, 31, 54]. The FSDP has been tied to problems with finite-size effects and the duplication of the long and intermediate-range order of silica in molecular dynamics simulations.

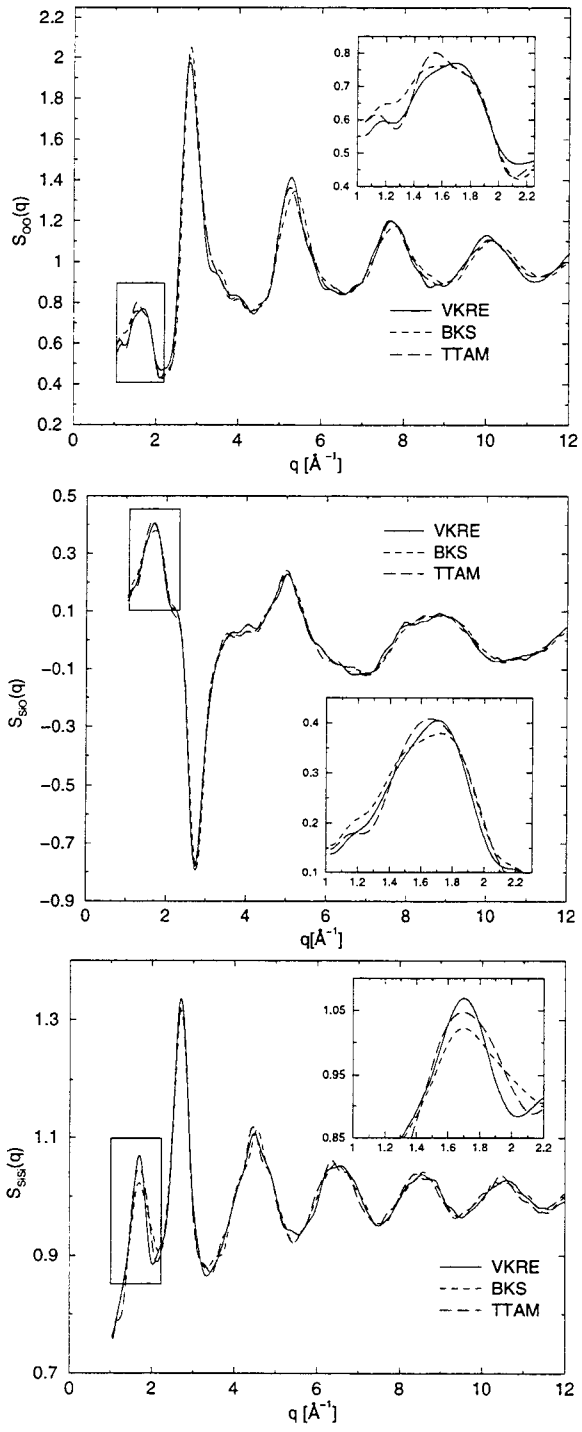

FIGURE 3.9. The partial structure factors at 0 K for each of the three interactions.

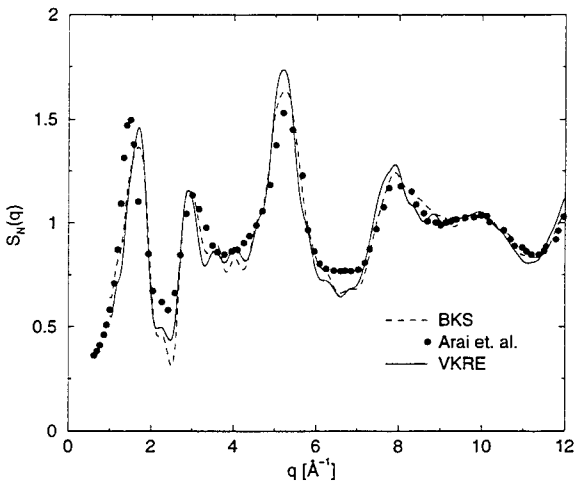

FIGURE 3.10. The total neutron weighted structure factor. The closed circles are neutron scattering data from Arai *et al.*

As Figure 3.9 shows, the main difference between the three structure factors is in the FSDP. There is variation in the peak heights in the Si-Si structure factor and in the shape of the peak for the O-O structure factor. There is much less variation in the Si-O structure factor. It is not clear whether the apparent differences in the first O-O peaks are real or are within the noise. However, it appears there is a significant difference in the peak heights of the first peak of the Si-Si structure factor. Regardless, there is no indication from these results to indicate whether or not the three-body interaction is preferable over the two-body interactions.

Figure 3.10 shows the neutron-weighted total structure factor for the BKS and VKRE interactions. The closed circles are experimental results from a neutron scattering experiment by Arai *et al.* [90]. The main differences between the simulation data and the experimental results appear in the first and third peaks. Both interactions overestimate the height of the third peak and both interactions

overestimate the position of the first peak. Again, from these results, it is not clear whether a three–body interaction performs any better or worse than a two–body interaction.

3.4.5. Density of States

The vibrational density of states was calculated from the velocity auto–correlation function [43],

$$\Gamma_\beta(t) = \frac{\left\langle \sum_{i_\beta} \mathbf{v}_{i_\beta}(t) \cdot \mathbf{v}_{i_\beta}(0) \right\rangle}{\left\langle \sum_{i_\beta} \mathbf{v}_{i_\beta}(0) \cdot \mathbf{v}_{i_\beta}(0) \right\rangle} \text{ with } \beta \in \{\text{Si}, \text{O}\} \tag{3.16}$$

where the angular brackets ($< ... >$) represent averaging over MD configurations. The partial vibrational density of states is then given by the Fourier transform of the velocity auto–correlation function,

$$F_\beta(\omega) = \int_0^\tau \Gamma_\beta(t) \cos(\omega t) \mathrm{e}^{-\gamma(t/\tau)^2} \, dt \, , \tag{3.17}$$

where a Gaussian window function with $\gamma = 1$ and $\tau = 0.5$ ps is used. The total vibrational density of states is obtained by summing over the partial density of states weighted with the concentration

$$F(\omega) = \sum_\beta c_\beta F_\beta(\omega) \, , \tag{3.18}$$

where c_β is the concentration of species β.

Figure 3.11 shows our results for the vibrational density of states for all three potential energy functions. Each graph is superimposed with data from a neutron scattering experiment from Price and Carpenter [91]. There is some high frequency noise that is created by the statistical noise in the data. We observe that this function is very sensitive to the choice of interaction. The primary features of the

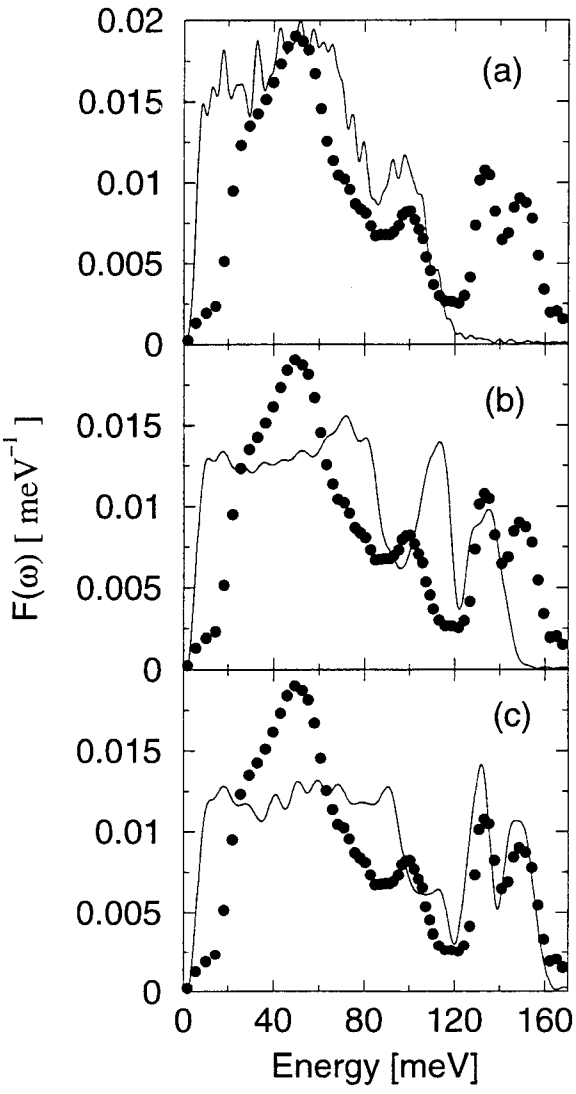

FIGURE 3.11. The vibrational density of states $F(\omega)$ for (a) the VKRE interaction, (b) the TTAM interaction, and (c) the BKS interaction. The closed circles are experimental data from Price and Carpenter.

TTAM and BKS results are a pair of peaks at the high energy end of the spectrum, and a fairly uniform and featureless plateau covering most of the low energy end. The experimental data has four obvious peaks at about 50, 100, 130, and 147 meV. The BKS results match the high energy peaks of the experimental data very well. The TTAM results have similar high-energy peaks, but they are shifted to lower energies as compared with experiment. Neither the BKS nor the TTAM results reproduces the peaks at 50 and 100 meV.

The VKRE results, on the other hand, appear to match the peaks at 50 and 100 meV, but show no evidence of any vibrational activity at energies above about 120 meV. It is possible that the high-energy experimental peaks at 130 and 147 meV exist within the VKRE spectrum and are shifted to lower energies. The integrals under all three of the molecular dynamics spectra are the same, indicating that no information is lost in the VKRE spectrum. It is clear that the VKRE interaction does not accurately reproduce the high energy peaks in the experimental spectrum of silica.

We hypothesize that this may be due to the fact that the Si-O attraction is much weaker in the VKRE interaction, which is mainly due to the lower effective charges of silicon and oxygen. The weakness of the attraction also makes the VKRE interaction much more diffusive than the BKS and TTAM interactions. Taraskin and Elliot [55] have shown that the two high energy peaks in the vibrational density of states correspond to stretching vibrations between silicon and oxygen pairs. Therefore a weaker interaction might shift these peaks to lower energies. This hypothesis is further strengthened when one considers that the TTAM and BKS interactions have equal effective charges on the silicon and oxygen ions, but the short range forces are slightly less attractive for the TTAM interaction. The TTAM density of states shows that the two high energy peaks are shifted to lower energies as compared with

the BKS interaction which indicates that this may be related to the strength of the Si-O interaction.

3.5. Conclusion

We have highlighted and demonstrated some differences in results obtained from the simulation of amorphous silica using two–body and three–body interactions. As expected, the three–body potential energy function more accurately reproduces the Si–O–Si bond angle distribution with respect to experimental results.

The VKRE interaction proved to be much more diffusive than the BKS or TTAM interactions. We discovered approximately 4-5 orders of magnitude difference between VKRE diffusivities and experimental results, and a maximum of two orders of magnitude difference between the two- and three-body interactions. This discrepancy is due to the comparatively low effective charges in the VKRE interaction.

All three interactions differ in the position of in the first sharp diffraction peak (FSDP) of the total static structure factor as compared with experiment. While the molecular dynamics results for all three agree on the position of this peak they differ with each other on its height. Overall, however, we observed quite good qualitative agreement with experiment for all three potential energy functions with respect to the total structure factor.

The most striking difference between the three potential energies was discovered in their vibrational density of states. The VKRE interaction was not able to produce the experimental peaks at 130 and 147 meV, while the BKS spectrum matched these peaks quite well and the TTAM results had two similar peaks that were shifted to lower energies.

It would be interesting for future studies to compare the results of the BKS and VKRE interactions in reproducing the density maximum of silica, and the IR band separation problem as documented by Hemmati and Angell [72].

We thank Dr. Priya Vashista at Louisiana State University. Thanks also to Dr. Walter Kob, and special thanks to Dr. C. A. Angell for informative discussions. We also thank the computer science department at Oregon State University for a considerable amount of computer time.

4. TABULATED POTENTIALS IN MOLECULAR DYNAMICS

David A. Wolff and Walter G. Rudd

To appear in *Computer Physics Communications*
Elsevier Science, in press

4.1. Introduction

Molecular dynamics (MD) [1–3] is a proven means for simulating systems of atoms, molecules, and other particles. The method involves integrating the equations of motion derived from inter-particle potential energy functions. Statistical averages then yield mechanical, thermodynamic, and transport properties of the simulated systems.

In MD, computing the inter-particle potential and the forces between the particles is by far the most time-consuming aspect of the method for all but the simplest of potentials. There are informal reports that looking up potentials and forces in tables stored in memory can yield significant improvements in performance in MD calculations [92], but we know of no published work that focuses on the idea itself. In this report, we describe our study of the effects of replacing computations of the potentials and forces with table lookups for these quantities.[1]

We focus on potentials that include terms that can be computed to any desired accuracy as part of the initiation before actual simulations begin, with emphasis on semi-empirical functions of relative positions of atoms, such as the Lennard Jones (LJ) potential. The potentials we consider are thus response surfaces on which the simulations are based. These can be indexed by functions of atomic positions. The most common use of this in practice has been in tabulating the "Embedded Atom Method" or EAM potential. There are MD codes in use that use tabulated EAM potentials exclusively [6, 7].

---

[1]Henceforth, we will use "potential" to represent potential energy, and we will use "atoms" to represent the atoms, molecules, or other particles that comprise the systems being simulated.

This is by no means the only way in which tables are used in MD calculations. For example, implementations of the MD-MC/CEM technique [93, 94] use tables to store pre-computed integrals indexed by local jellium densities and to hold coefficient matrices used in numerical differentiations of the potentials. Tabulated potentials have also been used to accelerate the calculation of long range electrostatic forces in the Ewald sum [37].

Memory cycle times on common workstations are about 60ns, which is approximately the time required for a single double-precision (DP) floating-point calculation. We therefore would expect that looking up values for potentials would be faster than computing them for all but the most trivial forms of the potential.

However, this is not always the case. Processors used in serious MD calculations include internal pipelined processors for floating-point arithmetic. Many offer vector processors as well. Both these processor enhancements require a steady flow of operands from memory. Efficient caching hardware and algorithms and optimizing compilers are tuned to prevent delays due to interruptions in the movement of data from memory. These techniques work best when the data access patterns are localized in memory and occur in a predictable order. On the other hand, using tabulated potentials is equivalent to performing random accesses into large arrays, so that accesses are neither localized nor predictable. We therefore expect and observe a trade-off between using tables to save arithmetic calculations and degradation in raw processor performance. Nevertheless, we expect to benefit from tabulating potentials for those that require relatively intense arithmetic computation and relatively little associated logic. This study supports that expectation.

Even when tabulating potentials offers better performance than computing them, there are still two other trade-offs to consider in deciding which method to use. The use of tables guarantees a loss of accuracy in computing the potentials. We

use DP arithmetic in MD simulations to eliminate possible inaccuracies that could cloud the interpretation of results. It is not clear that we need that level of precision in many simulations. Often we do not know values of parameters in potential functions with any degree of certainty; in much of the work in materials science and other application areas we use semi-empirical potentials that involve constants that are known to only one or two decimal places in accuracy. These values frequently appear in power-law or exponential terms in potential functions. Therefore, inaccuracies in determining values of potentials under these circumstances are swamped by lack of precision in our knowledge of the exact potential. Note that this comment does not apply to numerical techniques used to integrate the equations of motion; these techniques must be chosen carefully to ensure that they are sufficiently accurate. Otherwise, numerical instabilities can arise that lead to erroneous or divergent solutions.

The third trade-off between using tables and computing potentials and forces is the space-time trade-off, which is also linked with the accuracy issue. To make the computations faster, we must use memory to store tables that would otherwise be available for other purposes. This factor becomes particularly significant when we consider simulating systems with more than one species of atom. For two-body potentials, the number of tables increases with the number of possible pair interactions,

$$\binom{n}{2} + n \tag{4.1}$$

where n is the number of atomic species. When the calculations are to be done on parallel or distributed processors, we must keep copies of the tables on each processor to avoid extra inter-processor communications. Memory is an important factor in limiting the sizes of simulation we can do. As main memory sizes continue to grow,

the relative overhead from the tables becomes smaller. Our results described below indicate that under normal circumstances the use of memory for tables is not an important consideration.

In this report, we explore all of these issues. We begin with a discussion of how one might construct and access tables efficiently. We then present our results on the analysis of errors resulting from the use of tables and their effects on the results of simulations. Following a comparison of performance between the two methods for several different kinds of potentials, we present our conclusions and recommendations.

4.2. Implementing Tables

Well-designed molecular dynamics codes [2, 6, 7, 4, 5, 8] include single modules that compute potentials. To change a code to obtain potentials and forces from tables, we simply replace the code that computes these values with code that retrieves them from data structures in which they have been stored before the simulations are started. In order for this to be worthwhile, an efficient method of indexing the table must first be found.

4.2.1. Hash Functions

The objective is to determine the contributions to the force and potential acting on particle i by all the other particles j. We assume these forces depend only on separations r_{ij} between the particles. The potential/force routines in MD systems usually include checks to see if particles j are within a certain distance from particle i, as for example, when the potential function includes a short-range cut-off. These comparisons are done using r_{ij}^2 to avoid computing a square root. Therefore,

it makes sense to use r_{ij}^2 as the measure of the distance used to determine which elements of the tabulated forces and potentials to use.

Since pointers and indexes for arrays are integers, we need to construct mappings between double precision measures of (squared) distances and pointers or indexes. We have investigated two kinds of mappings, casts of scaled values of the distance measure onto integers, and logical extraction of indexes from the DP distance measures.

Consider the pseudo–code below for the computation of the potential and force in a MD simulation.

for each r_{ij}^2 **do**

$e :=$ computeEnergy(r_{ij}^2)

$f :=$ computeForce(r_{ij}^2)

totalE := totalE + e

$\mathbf{F} := \mathbf{F} + f\hat{r}_{ij}$

Note that this is for a two–body potential. For a three–body potential the loop would be over triplets (r_{ij}^2, r_{ik}^2, and θ_{ijk}). The computeEnergy and computeForce routines in practice are not separate routines, but since some of the computation is redundant, they are often just one function or are inlined directly into the loop.

The table version of the above code might look like the following.

for each r_{ij}^2 **do**

$e :=$ energyTable[hash(r_{ij}^2)]

$f :=$ forceTable[hash(r_{ij}^2)]

totalE := totalE + e

$\mathbf{F} := \mathbf{F} + f\hat{r}_{ij}$

Here, hash() is a function which maps pairwise distances to indexes in the tables. An example is a function that simply casts DP values to integers:

```
function hash(double t)
  return (int) ( (t) * 10000 ) )
```

This yields the integer part of the argument after shifting the decimal point four places to the right.

We could simulate a rounding operation with the following hash function.

```
function hash(double t)
  return (int) ( (t * 10000 ) + 0.5 )
```

The above two mappings are simple, but require converting a DP number to an integer, a process that is not inexpensive.

4.2.2. Double Precision Extraction

The other method we consider for determining indexes into potential arrays is the extraction of an index directly from a DP representation of some measure of distance between atoms. The advantage of this technique is that it eliminates the arithmetic, logic, and implicit function calls involved in converting DP values to integers.

For a given double precision exponent, we can use the mantissa, truncated at some precision, as an integer to access the table. The table size must be of size at least 2^{N_m} where N_m is the number of bits taken from the left-hand side of the mantissa.

However, in most cases we are interested in a range of numbers which include several different possible double precision exponents. In this case, we can use a few of the low order bits in the exponent as the first few high order bits in the integer (see Figure 4.1). Of course, care must be taken to make sure that enough bits are taken from the exponent to ensure that they are unique for all possible exponents that would be encountered in a simulation.

We will assume that a double precision number is 64 bits long. A function for extraction of the INDEX_BITS of Figure 4.1 might be the following (the code is written using C syntax, and line numbers are inserted for future reference).

```
int extract(double t) {
    int *int_p;
    int index;

(1) int_p = (int *) &t;
(2) index = ( *int_p & MASK );
(3) index = index >> ( 20 - MANTISSA_BITS );
    return( index );
}
```

We have shown extract() as a function and, for clarity, we have written the computations in several steps. In practice, it should be in–lined within the code in order to avoid the overhead of the function call.

In the extract() function, MANTISSA_BITS is the number of bits of the mantissa which we would like to use. In other words, we are truncating the number at a precision of $2^{-\text{MANTISSA\_BITS}} \times 2^x$ where x is the value of the exponent. The total

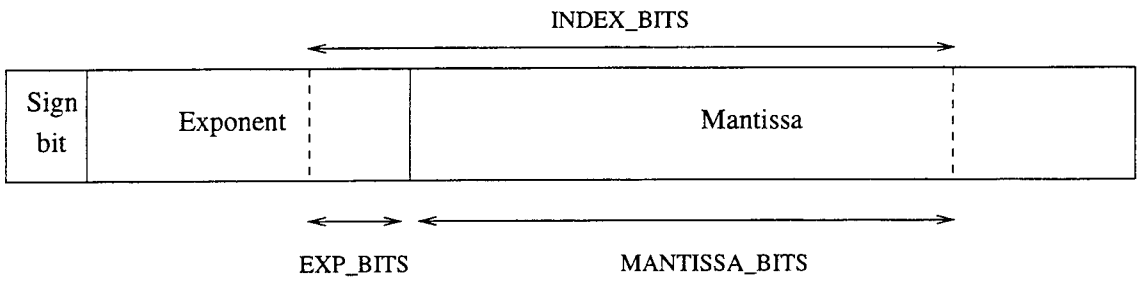

FIGURE 4.1. Representation of double precision index extraction. INDEX_BITS are the bits that are actually extracted to index the table.

length of an index to the table is INDEX_BITS, which is equal to MANTISSA_BITS plus the number of bits taken from the exponent (EXP_BITS in Figure 4.1). The value of INDEX_BITS determines the sizes of the tables, which should be of length at least $2^{\text{INDEX\_BITS}}$.

It is important to note that the table size is determined solely by the length of the integer used to access it. However, all of the table will not necessarily be used. The entire table will be used only when given a set of possible double precision inputs, INDEX_BITS includes all possible binary strings. Clearly, this depends on a number of factors including whether or not all possible binary strings are expected in the EXP_BITS, and whether or not the full range of mantissas for a given exponent is expected.

Here are what the numbered steps in extract() do:

1. We establish a pointer to an object of type int and set it to point to the DP argument. This allows us to ignore the type rules that ordinarily would restrict us to using DP operations on the argument. Since a integer is typically 32 bits and a double is usually 64, we are restricted to the leftmost 32 bits of the

double. With an exponent of 11 bits, there are 20 bits of the mantissa to work with. One could use two integer pointers to point to the first and second half of the double in order to use more bits of the mantissa. However, this requires some extra instructions to combine the two.

2. MASK is used in an *and* operation (&) to remove the left hand bits from the argument. MASK has the value $2^{\text{INDEX\_BITS+20-MANTISSA\_BITS}} - 1$.

3. Shift the resulting integer to the right to produce an index of the required length.

As an example case, consider indexing a table when our pairwise distances r_{ij}^2 range from 0.5 to 127.9 Å^2. This would be reasonable for a short range pair potential with a cutoff of about 11.3 Angstroms. The exponents of the floating point representations range from -1 to 6. In bias 127 or bias 1023 notation these exponents are unique in their three lowest order bits. Not only are they unique, but for this range of values they complete all possible three bit strings. Hence, if we create a table using three as the value for EXP_BITS and say 17 for MANTISSA_BITS, we can create a table of size 2^{20} with nearly every entry filled. We would also be accurate in the value of r_{ij}^2 to an precision of about 2^{-17+6} in the worst case.

The key here is to make the mapping function as easy to compute as possible. While it appears that the extract() function is more complex than the hash() function, the fact is that the integer cast is an expensive operation. We have found the hash() function requires about 5 times as much time as the extract() function.

4.2.3. Multiple Tables

MD codes often deal with more than one species of atom, making it necessary to use more than one table. This can be done quite readily for example by making an array of tables. The first two dimensions of the array might be indexed by a code for the species of the atom, and the third dimension would be the table itself. We would arrange the pointers in the top two levels of the array so that an interaction between atoms would access the same table, independent of the order in which the species are specified. An access to the table might be the following:

e = tables[type1][type2][hash(r_{ij}^2)];

where type1 and type2 refer to the atomic species.

4.2.4. Interpolation

Finally, one could interpolate between successive points in the table for more accurate results for the forces and potentials. A linear interpolation is most often all that is required, although, more complicated interpolations have been used [95]. The same effect can often be achieved at little extra cost by increasing the size of the tables.

4.2.5. Constructing Tables

Construction of the table is done before the simulation starts. We have found that for realistic problems computing the tables takes a negligible amount of time, and so we compute them with each run. It would make sense to store tables that are expensive to compute on disk.

Here is sample code to compute a table.

$r := r_{\min}$

while $r \leq r_{\max}$ **do**

rsqr := $r * r$

index := hash(rsqr)

energyTable[index] := computeEnergy(rsqr)

forceTable[index] := computeForce(rsqr)

$r := r + \Delta r$

The routine above builds a table for $r_{\min} \leq r \leq r_{\max}$ where $r_{\max}$ is the cutoff distance for the potential, and $r_{\min}$ is some minimum distance where it is known that two atoms will never be less than $r_{\min}$ apart. This, of course, depends on the details of the simulation.

When using the hash functions supplied above, some of this table would be left unused because the r^2 distances that map to the lower indices to the table $r < r_{\min}$ will never be needed. This is also often (although not necessarily) the case with the DP extraction routine. Depending on the size of the table that is needed the wasted space my not be a large concern. One could modify the hash function so that no space is wasted, but that would be at the cost of additional arithmetic when accessing the table.

It is possible to use hash functions along with chaining methods for collision resolution to help to eliminate the wasted space. We experimented with this approach and determined that the extra coding and computational effort required probably does not justify the small savings in memory.

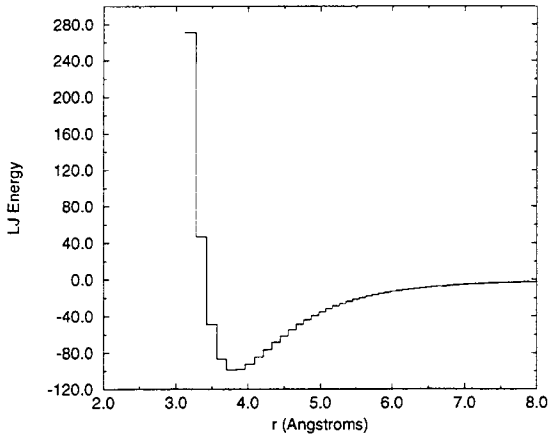

FIGURE 4.2. The Lennard-Jones table-based potential. The spacing of the table values is much larger than would normally be used in practice.

4.3. Errors and Error Accumulation

The use of tables without interpolation changes the potentials and force functions to sequences of step functions (see Figure 4.2). Again, at the cost of some arithmetic, we can improve the accuracy of tabulated potentials via interpolation. In our codes, we store both the potential and the force functions so that we need not differentiate the potential function numerically to obtain forces.

4.3.1. Estimation of Error in the Potential

In order to determine a more quantitative picture of the errors introduced by the use of a tabulated force field, we consider a general example. Most of the common potentials in use today have terms that depend either on an inverse power

of the distance ($1/r^p$) or exponentially on distance (e^{-r}). We will consider both cases.

First we express the relative error for an inverse power term in a single table look–up as,

$$\varepsilon_\phi = \left| \frac{\phi(r) - \phi^{\text{table}}(r')}{\phi(r)} \right| = \left| \frac{\frac{1}{r^p} - \frac{1}{r'^p}}{\frac{1}{r^p}} \right| = \left| 1 - \frac{r^p}{r'^p} \right| \tag{4.2}$$

where r is the actual position of the particle and r' is the truncated or rounded value used for a table lookup.

We can access the table by either rounding the value to the nearest table entry, or by simply truncating (rounding down). In the case of the rounding method of accessing the table, because of the increasing, negative slope of $\frac{1}{r^p}$, the maximum possible error occurs when $r < r'$ and $r' - r = \Delta r/2$ where Δr is the table spacing (see Figure 4.3). Hence,

$$\varepsilon_\phi \leq \left| 1 - \left(\frac{r' - \frac{\Delta r}{2}}{r'} \right)^p \right| \tag{4.3}$$

Since p is positive and r' is always greater than $\Delta r/2$ we can write the upper bound as

$$\varepsilon_\phi \leq 1 - \left(1 - \frac{\Delta r}{2r'} \right)^p \tag{4.4}$$

This gives an upper bound for the error in $1/r^p$ type potential. The maximum value that ε_ϕ can take on depends on the smallest possible value for r' for a given Δr. To determine this we must look to the particular substance being simulated. The pair distribution function $g(r)$ for a substance gives us information on the environment of a given atom. Since most atoms act like hard spheres for small

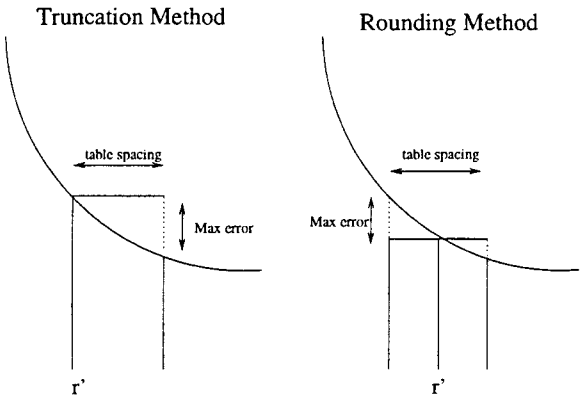

FIGURE 4.3. Representation of a single table entry. Note that the error is largest when $r = r' + \Delta r$ for truncation, and when $r = r' - \Delta r/2$ for the rounding method.

values of r, the pair distribution function must go to zero at some small value of r, lets call it $r_{\min}$. Hence, we can expect that no atom will ever be within $r_{\min}$ of another. This gives us a lower bound on r' in Equation 4.4. Hence we can write the maximum possible error in the potential as

$$\varepsilon_{\phi}^{\max} = 1 - \left(1 - \frac{\Delta r}{2r_{\min}}\right)^{p}. \tag{4.5}$$

For our argon case study, the pair distribution function goes to zero at about 3.0 Å. So for a Δr of 0.001 Å and $p = 6$, $\varepsilon_{\phi}^{\max} = 0.001$. This is respectable when compared with the accuracy to which many potential parameters are known. Also, it should be noted here that this may also be on the order of the accuracy given by the integration method used. Most integration methods conserve total energy to within the order of the timestep size. And perhaps more importantly, most table lookups will have substantially smaller error. In addition, for the rounding method, we can err on either side of the actual value for the potential (Figure 4.3). That is, the

tabulated value might be larger or smaller than the actual value. Hence, we can expect that some of the overall error will be canceled out by this effect.

A similar analysis can be done for an exponential potential,

$$\phi(r) = \mathrm{e}^{-\mathrm{r}} \tag{4.6}$$

which yields,

$$\varepsilon_\phi \leq \left| 1 - \mathrm{e}^{-\frac{\Delta \mathrm{r}}{2}} \right|. \tag{4.7}$$

4.3.2. Accumulation of Errors

For a pair potential, the total potential energy for a single timestep is simply a sum of pairwise potential values,

$$\Phi = \sum_{i<j} \phi(r_{ij}), \tag{4.8}$$

where,

$$r_{ij} = |\mathbf{r}_i - \mathbf{r}_j| \tag{4.9}$$

is the distance between atom i and atom j. For the case of the exponential potential, the relative particle–particle error is independent of the pairwise distance. Equation 4.5 gives an expression for the $1/r^p$ potential that is also independent of the particle–particle distance.

Now, since the maximum relative error of each term in the sum is independent of the pairwise distance, the relative error in the sum is the same as the relative error of each term. Hence, Equations 4.5 and 4.7 also hold for the total potential,

$$\varepsilon_\Phi = \varepsilon_\phi. \tag{4.10}$$

Next, we consider the total force on a given atom,

$$\mathbf{F}_i = \sum_j f(r_{ij})\hat{r}, \tag{4.11}$$

where

$$f(r) = \frac{\mathrm{d}}{\mathrm{dr}}\phi(r). \tag{4.12}$$

It can be shown that the relative error in $f(r)$ for the $1/r^p$ potential type is the same as Equation 4.4 except with an exponent of $p+1$. Similarly, it is easy to show that the relative error in $f(r)$ for the exponential type potential is the same as Equation 4.7.

Now, the force is a vector quantity, so the summation is a little different here as opposed to the total potential energy. However, one can still put an upper bound on the total error by assuming that all of the errors point in the same direction. Again the same argument applies as above. The relative error in the force for a single particle is

$$\varepsilon_{|\mathbf{F}_i|} \leq 1 - \left(1 - \frac{\Delta r}{2r_{\min}}\right)^{p+1}, \tag{4.13}$$

for the $1/r^p$ potential. For the exponential potential a similar analysis reveals the same upper bound as Equation 4.7.

4.3.3. Errors in Measured Values

Now that we have an upper bound on the relative error for the force and potential for a given timestep, let's consider how those errors propagate into some of the quantities that are measured. The pressure, temperature, stress, and total energy all depend on various summations involving the force, positions, and velocities of the particles. We have already discovered how the errors in individual table lookups propagate to the total force and potential energy.

The total energy is given by

$$E = \sum_{i<j} \phi(r_{ij}) + \frac{1}{2} \sum_{j} m_j v_j^2. \tag{4.14}$$

We have already put an upper bound on the error in the first term, it remains to put an upper bound on the error in the second.

The velocity is obtained from the total acceleration by integrating over some short time interval. There are many methods of doing this, [1], but for simplicity and in order to consider upper bounds, we will consider just a Taylor expansion of the velocity:

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \delta t \mathbf{a}(t) + \cdots . \tag{4.15}$$

For a given timestep there is some error in the force (acceleration) due to the use of a tabulated force field. This is equivalent to adding a term to the acceleration,

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \delta t(\mathbf{a}(t) + \varepsilon_a \mathbf{a}(t)), \tag{4.16}$$

where ε_a is the relative error bound in the acceleration which is equal to the relative error bound in the force. Now, this term is of order $\delta t \varepsilon_a a$ which is of similar order as the error in the potential energy term. In the expression for the total energy, this term is squared, which makes the error of order $\delta t \varepsilon_a \mathbf{a} \cdot \mathbf{v}$. Regardless, we can consider the error in the total energy to be proportional to ε_Φ. We use ε_Φ instead of ε_a because the binomial is raised to a higher power in ε_a. In any case, as will be seen in the next section, we are most interested in how the error changes with respect to the table size, and for this purpose, the above argument is sufficient.

4.3.4. Error With Respect to the Table Size

We expand the binomial in Equation 4.4,

$$\varepsilon_\phi \leq p\frac{\Delta r}{2r} - \frac{p(p-1)}{2}\left(\frac{\Delta r}{2r}\right)^2 + \cdots, \tag{4.17}$$

where we have dropped the prime on r. Since, $\Delta r \ll 2r$ for all reasonable[2] r, we know that

$$0 < \frac{\Delta r}{2r} \ll 1. \tag{4.18}$$

Now, we would like to truncate the series to first order, but we must first convince ourselves that the upper–bound still holds in that case. In other words we must show,

$$1 - (1-x)^p \leq px, \tag{4.19}$$

for $0 \leq x \leq 1$. Since $p \geq 1$, this is true for $x = 1$ and for $x = 0$. Consider the first derivative of both sides with respect to x,

$$p(1-x)^{p-1} \leq p. \tag{4.20}$$

Since we are only concerned with the range $0 \leq x \leq 1$ and since $p \geq 1$, this is clearly true. Having established that the slope of the left hand side is always smaller or equal to the slope of the right hand side, and that the endpoints of our range of interest satisfy Equation (4.19), we can say that Equation (4.19) holds for all $0 \leq x \leq 1$.

---

[2]By reasonable r, we mean any value of r which will normally be encountered in a simulation. This is bounded at the low end by the point at which the pair distribution function $g(r)$ becomes negligible.

Truncating Equation (4.17) to first order in Δr, we have:

$$\varepsilon_\phi \leq p\frac{\Delta r}{2r} \leq p\frac{k}{2rN}, \tag{4.21}$$

here $\Delta r = k/N$ where $k = r_{\max} - r_{\min}$ is the range of values of r in the table, and N is the number of table entries or the size of the table.

This indicates a linear relationship between the inverse table size and the relative error.

4.3.5. Errors in Practice

In practice, the upper–bounds that we have derived in the previous sections are very high. In fact the actual errors that are incurred overall will be of a much lower magnitude. This is because of several reasons.

First, the error in a table lookup can be on either side of the actual value. Hence, we can expect some "washing out" of the errors when multiple values from the table are summed. In fact, the error will act as a one- (for potential energy) or three-dimensional (for the force) random walk. So the absolute errors would sum as order $\sqrt{N}$ where N is the number of terms in the sum.

Second, at least for $1/r^p$ potentials, the error is reduced for atoms that are farther apart than $r_{\min}$. As was stated previously, $r_{\min}$ is the minimum distance that two atoms might approach each other. This is indicated by the point at which the pair distribution function for the substance being simulated goes to zero.

Figure 4.4 shows a graph of the pair distribution function for liquid argon overlaid with the maximum error as a function of r. The region of largest error is where the potential is steepest. Also included in the graph is the product of the two functions. The pair distribution function $g(r)$ is proportional to the probability of

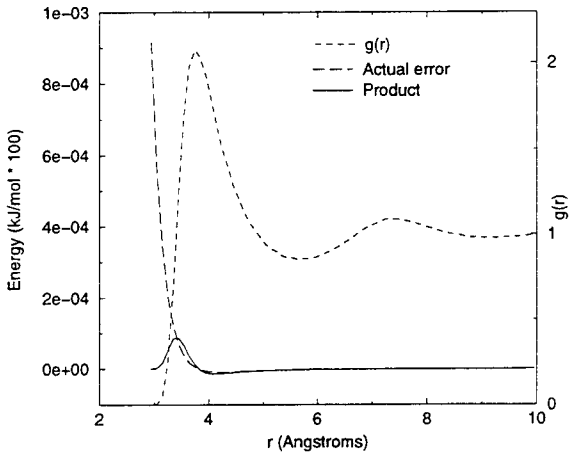

FIGURE 4.4. The scaled, actual error $(\phi(r) - \phi(r + \Delta r))/\epsilon$ and pair distribution function taken from an MD simulation of LJ liquid argon, where ϵ is the depth of the potential well. The solid line is the product of the error and the pair distribution function.

finding an atom at a distance r around any given atom. The product of the two functions shows a maximum at a point slightly to the left of the maximum in $g(r)$, and goes to zero with $g(r)$. This demonstrates that distances between neighboring atoms are seldom in the region of largest error, and that the vast majority of the table lookups will be in the range ($\geq$ 3.6 Å) where the error is very small.

Finally, for purposes of analysis we have made the simplification that the potential energy is made up of a single term that is always positive. In reality, the potential energy will be a sum of positive and negative contributions from these terms. In such cases we would need to be much more careful in using the relative error since the potential energy and the force may be zero. Nevertheless, we can still do a similar analysis on each term of the potential and then combine the errors

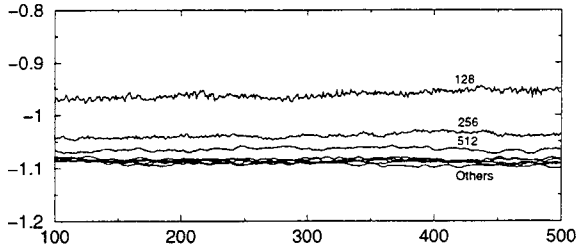

FIGURE 4.5. Total energy per atom *vs.* time for several sizes of potential tables.

after the summation. An alternative method is to look at the absolute error and take into account the "random walk" nature of the summations.

4.4. Experiments on Sample Systems

4.4.1. Variation of Table Size

Figure 4.5 shows the results of tests in which we varied the sizes of the potential and force tables. Each curve is a 50,000-step run for LJ liquid argon. The systems consisted of 10,648 atoms. We used Rapaport's code [2] and a leapfrog integrator with reduced time steps of .001.

The table sizes are determined by the parameter `INDEX_BITS` used in the extract() mapping described above. We see that results using tables of sizes 128, 256, and 512 elements for each table move toward the aggregate of results using larger tables, labeled "Others" in the graph, which are essentially indistinguishable. The larger tables contained 2^n elements, for n = 10, 12, 14, 16, 18, and 20.

We computed the average energy for the last 25,000 timesteps for each table size in Figure 4.5. Taking the average value of the three curves (n = 16, 18, and 20) which are indistinguishable as the "exact" energy, we plotted relative error

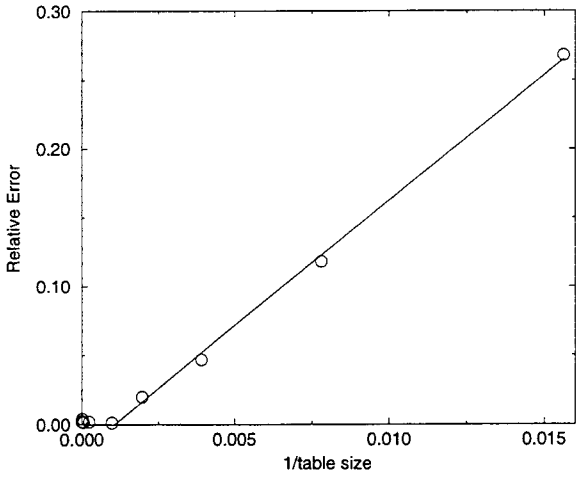

FIGURE 4.6. Relative error in total energy *vs.* inverse table size for the experiment in Figure 4.5.

versus $1/N$ (where N is the table size) in Figure 4.6. This demonstrates the linear relationship predicted in Equation 4.21. Hence, we can see that the expression for single particle error contributes in a similar way to overall errors in the energy. The fact that the least-squares linear fit does not pass through the origin is representative of other errors that do not depend directly on the table size.

These results indicate that tables of size 2^{12} or larger are more than adequate (at least for the total energy); increasing the table size beyond that has no noticeable effect on the results for the total energy. This also indicates that interpolation may not be necessary.

We obtained similar results in simulations of 216 silicon atoms in the liquid state using a table for the pair part of the Stillinger-Weber potential [32].

One would expect that so long as the tables are large enough to provide the same results as computed potentials, it makes no difference how large the tables are. However, to do so would be to ignore the cache effect. In the LJ argon case, for example, we found that the time per call to the routine that computes all the forces in a time step increased by 50% when n was increased from 14 to 16, which represent table sizes of 16K and 64K entries, respectively, on an ultrasparc processor with a 1 Mbyte cache. Storage requirements for these tables were 256K and 1M bytes respectively. There was no further increase in execution time for larger tables up to 2^{20} entries.

4.4.2. Comparison of Output

We now describe our comparisons of the results from simulations that use tabulated potentials and simulations in which the potentials and forces are computed with every step.

Table 4.1 shows the results for LJ liquid argon using the Moldy [4] code. We note that, despite the relatively short times used for equilibration and for accumulating averages, the variations within energy are well within acceptable limits and the standard deviations in energy and temperature show the expected behaviors as the size of the system increases.

Figure 4.7 shows the simulation time dependence of the total energy per particle for LJ liquid argon using the two methods. The simulation was for 50,000 steps for 10,648 atoms using the Rapaport code. While the fluctuations in energy between the two simulations are not synchronized after about 10,000 steps, there is no significant difference between them.

Number of atoms	Energy (kJ/mol)	Std. Dev.	Temperature (K)	Std. Dev.	Type
100	-152.64	0.27	140.53	5.53	Computed
100	-152.75	0.28	140.67	5.52	Tabulated
200	-275.85	0.42	150.38	4.06	Computed
200	-275.97	0.41	150.52	4.06	Tabulated
500	-785.90	0.74	140.28	2.57	Computed
500	-785.73	0.72	140.09	2.48	Tabulated
1000	-1715.21	1.11	132.87	1.73	Computed
1000	-1715.05	1.08	132.85	1.78	Tabulated
5000	-8447.27	2.43	134.13	0.80	Computed
5000	-8448.92	2.55	134.22	0.82	Tabulated
10000	-15930.54	3.38	139.34	0.54	Computed
10000	-15931.17	3.17	139.33	0.57	Tabulated

TABLE 4.1. Results for table and non-table based runs of liquid argon (density 0.8 g/cm^3 and initial temperature 120 K) with the Lennard–Jones potential. Averages are over 45000 timesteps after 5000 timesteps of equilibration with a timestep size of 0.01 ps. No scaling of the temperature is performed.

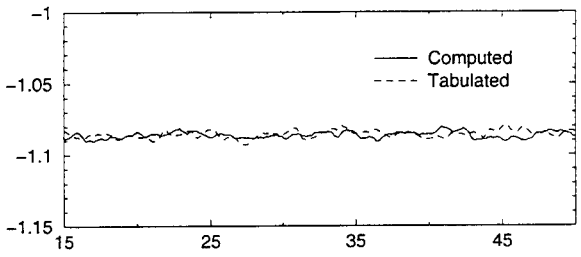

FIGURE 4.7. Total energy per atom using computed and tabulated potentials and forces for a LJ argon system.

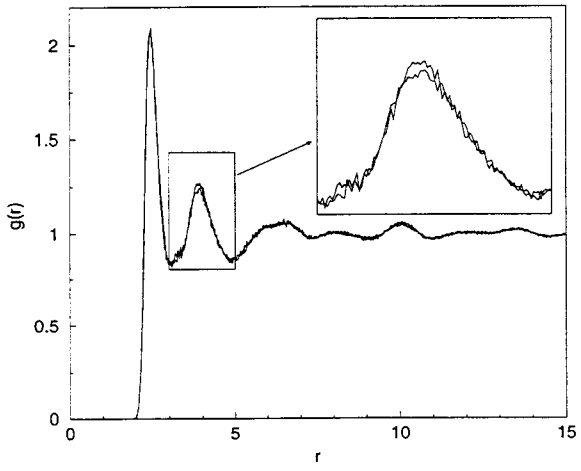

FIGURE 4.8. Pair-correlation function from computed and tabulated potentials and forces for a liquid silicon system.

Figure 4.8 shows that there is no significant difference in structure between the results produced by the two methods. The inset shows a small difference in the heights of the second peak. This is close to being within the noise of the function, but could be significant for detailed structural studies. We conclude that, at least for the equation of state and structure studies in which general structural information is needed there is no important difference in results between the two methods.

4.5. Performance

We turn now to an analysis of the relative performance of the two methods.

We have focused on three forms for potentials: (1) polynomials in $1/r$, such as the LJ potential, (2) "generalized" potentials, which are combinations of polynomials and exponentials, and (3) three-body potentials, such as the Stillinger-Weber potential for silicon. The latter class involves considerable logic and arithmetic to compute the three-body contribution to the potential in addition to the polynomials and exponentials in the two-body contribution in the potential.

Table 4.2 shows the performance results for the LJ potential. The simulations were on a system of argon atoms. Kernel times refer to the amount of time spent in the subroutine that calculates the potential and force. The relative performance is reproducible across codes and machines. We used the Rapaport code and Moldy [4] on HP, Sun, and DEC workstations, with little variation in differences in relative performance.

These results might seem disappointing at first, since it would seem that the LJ potential would require a moderate amount of arithmetic. However, a well-coded LJ calculator requires only three multiplications and a subtraction, if r^2 is already available, and a good compiler can rearrange the loop through the atoms and manage

Experiment	Time in Kernel (s)	Time per force call (ms)
Table, 2^7 elements	877	14.9
Table, 2^{14} elements	946	16.3
Table, 2^{20} elements	1,619	29.5
Computed	998	17.3

TABLE 4.2. Kernel time and time per call to the force routine for the LJ liquid argon experiments.

the cache to make sure that the pipeline is always full. Furthermore, on modern processors floating-point arithmetic is often faster than the integer arithmetic and random logic our extract() routine executes. Hence, one would expect to see very little or no improvement with tabulated potentials for a LJ potential.

Table 4.4 shows the results for a generalized potential with parameters for liquid argon. The generalized potential has the following form.

$$\phi(r) = Ae^{-Br} + \frac{C}{r^{12}} - \frac{D}{r^4} - \frac{E}{r^6} - \frac{F}{r^8} \tag{4.22}$$

This experiment was done by inserting a table into a previously established MD code. The code that we used was called "Moldy" [4]. The timings for the kernel routine show that the table method is about a factor of four faster (in kernel time) than direct computation. Our tests on one of the machines on which these calculations were done, a 170 MHz ultrasparc, showed that floating-point multiplications, square roots, and transcendental function evaluations require .006, .21, and 1.8 μseconds, respectively. The exponential functions make the difference.

Time in kernel	Total Time	Seconds per Timestep	Type
370.80	1529.64	0.77	Tabulated
639.15	1810.45	0.91	Computed

TABLE 4.3. Timings for table and non table based runs of a SiO_2 system with 192 silicon atoms and 384 oxygen atoms. We see an improvement in speed of about a factor of 1.18 in execution time and 1.72 in kernel time. (Evaluation of the Coulomb interaction was omitted for the purpose of this study.)

The speedup is considerably less when one considers total execution time. This is because the computation of the potential is only one part of the computations that need to be done in a given timestep. Integration of the force, measurement of different parameters, movement of atoms between cells, and occasional neighborlist calculation make up the rest of a single timestep. For example, consider the speedup of a given timestep when the speedup in the kernel is 4,

$$\text{Speedup} = \frac{t_{\text{other}} + t_{\text{kernel}}}{t_{\text{other}} + \frac{1}{4}t_{\text{kernel}}}. \tag{4.23}$$

If the other calculations in a timestep are equally expensive as the kernel computations then we have,

$$\text{Speedup} = \frac{t_{\text{kernel}} + t_{\text{kernel}}}{t_{\text{kernel}} + \frac{1}{4}t_{\text{kernel}}} = \frac{8}{5}. \tag{4.24}$$

So, overall we would see a speedup of only 1.6 as opposed to 4.

Table 4.3 shows times for simulations of two species of atoms, which requires 3 tables. Here we used a potential of the 6–exp form

$$\phi(r) = -\frac{A}{r^6} + Be^{-Cr}, \tag{4.25}$$

which is similar to the generalized potential. When one compares the timings for this potential to the timings for the Lennard–Jones potential the significance of the exponential term with respect to speedup becomes clear. We observe that the results at first glance do not indicate a speedup as large as expected, since with one or two more multiplications we would have the generalized potential for which we measured a factor of 4 speedup. This is because there is now more than one table, and extra logic is included in the table lookup stage to determine which table to use. The reason this is necessary is due to specifics of the Moldy code. Hence, some optimizations by the compiler for the innermost loop are not implemented, not to mention the additional instructions needed to execute conditional statements. Nevertheless, the speedup demonstrates the significant contribution of the exponential.

Table 4.5 shows the performance of the methods on silicon, 4,096 atoms for 50,000 time steps using the Rapaport code. In generating the code for the force calculations, we were able to tabulate only the two-body terms. The three-body terms, as usually written, require the calculation of the cosine of the angle between each triplet of atoms that are within a cut-off distance of each other. The cosine is a dot product between two of the vectors linking the atoms. There is little point in constructing tables for these because they would require a table indexed by r_{12}, r_{13}, and $\cos(\theta)$. Hence, the table would have to be three dimensional which would increase the storage requirements by about a power of three. Also, very little arithmetic is saved since much of the work is in doing the dot product.

Nakano *et al.* [16] have derived a means for separating the three-body force into sets of two-body interactions. It would be interesting to see the gains that can be made through using their technique with a tabulated potential.

Number of Atoms	Time in kernel	Total time	Time per timestep	Type
10,000	2969.28	24759.33	4.95	Tabulated
10,000	10944.76	33467.90	6.69	Computed
5,000	2204.53	17783.34	3.56	Tabulated
5,000	7775.09	23188.80	4.64	Computed
1,000	370.91	3310.07	0.66	Tabulated
1,000	1617.31	4436.66	0.89	Computed
500	203.49	1463.60	0.29	Tabulated
500	689.00	1869.68	0.37	Computed

TABLE 4.4. Timings for table and non table based runs of liquid argon using a generalized potential. The kernel refers to the subroutine that calculates the potential and force for a list of pairwise distances. All times are in seconds.

Experiment	Milliseconds per Force Call per Atom
Table, 216 atoms, 5000 steps	.79
Table, 4,096 atoms, 50,000 steps	.75
Computed, 216 atoms, 5000 steps	.98

TABLE 4.5. Kernel time and time per call to the force routine for the silicon experiments.

Potential	Compute/Table time
Lennard-Jones	1/1
6–exp	7/4
Generalized	4/1
StillingerWeber	5/4

TABLE 4.6. Approximate ratios for computing potentials versus table lookups for three classes of potential.

4.6. Conclusions

In Table 4.6 we provide approximate ratios of times in force calculations for computed and tabulated potentials. One message is clear. One should certainly use tables for potentials when the potential involves transcendental function calls and other intensive computations with relatively little logic. The case for more complex potentials, such as three-body potentials and problems that require multipole expansions is still cloudy.

We have shown that the use of tabulated potentials can be effective in speeding up MD simulations. We have outlined different tradeoffs that need to be taken into account before implementing a table into a particular code. The number of different species to be simulated, complexity of the potential, and cache size are all important considerations. We have also investigated the errors introduced by tabulated potentials. Surprisingly small tables of a few thousand elements produce certain results that are indistinguishable from those obtained using double precision arithmetic.

5. A STUDY OF MULTI-THREADED TECHNIQUES IN MOLECULAR DYNAMICS

5.1. Overview of Threads

A thread [96, 97] is a series of instructions within a process, which can be executed as a program. Typically, a process contains only one thread which runs sequentially, sharing the processor with other processes via time–slicing or some other scheduling scheme. Multi-threading allows a single process to employ multiple, simultaneous threads which all share the process's memory, files, and other resources. These threads run independently of the others as if they were independent processes, while sharing the memory space of the parent process. In fact, a multi-threaded program can be thought of as many singly threaded programs running in separate UNIX processes, but sharing a memory space. On multiprocessor machines, the threads can be scheduled by the operating system to run on separate processors. Thus, multi-threading offers a simple way to parallelize programs for use on the shared memory multiprocessor machines that are sometimes called SMPs (Symmetric Multi-Processors).

5.2. Goals of Threading Molecular Dynamics

There would be no reason to use threads for molecular dynamics unless there is a performance increase. First, threads can help us to take advantage of shared memory multiple processor machines. In the following, we consider other possible benefits.

5.2.1. The Memory Bus Bottleneck

A molecular dynamics program is strictly a numerical procedure, or what is sometimes called a compute-bound program. That is, it seldom has to access the disk or other devices while the program is running. Of course, it still has to access main memory. The problem is that processor speeds are increasing at a much faster rate than speeds of the main memory. For example, a typical modern machine might have a processor that has a clock speed of about 400 MHz. This translates to an instruction cycle time of 2.5 ns. Typical memory speeds are about 50 ns, so the processor can compute 20 times faster than it can move the data to and from main memory. Hence, when data needs to be read from main memory, computation is stalled to wait for the data to be retrieved and be sent through the memory bus to the processor. Modern machines have caches to deal with just this problem. Nevertheless, cache sizes are seldom large enough to contain all of the data necessary for a typical molecular dynamics simulation. Thus, a molecular dynamics program spends some fraction of its total simulation time waiting for the memory bus.

It has been postulated that multi-threading could address just this problem [98]. If the simulation is broken up and given to separate threads, then when one thread is stalled waiting for data from the main memory bus, another thread could take over the processor.

However, threads are not currently implemented in such a way as to have a method for making a context switch (switching between threads) while waiting for data from the main memory. Such context switches do occur in conjunction with more expensive calls to I/O devices such as the hard disk, printer, or other such devices. Even if there was a mechanism for a context switch while waiting for

main memory, there are several things involved in the context switch which may nullify the speed gain. The thread scheduler must save the "state" of the thread – represented by the program counter, stack pointer, registers, *etc.* – so that it can be started again at the same point where it left off. Now, in the process of doing this write to memory, we are taking up the extra time that we had hoped to save in the first place! In other words, we are preempting a write (or read) to main memory with another write to main memory.

The question still remains as to whether these writes are to cache memory or not. One might expect a speed increase if all of the register saves were to cached memory. However, it is difficult to guarantee that this will always happen, especially with the amount of data that is being swapped into and out of the cache during a MD simulation.

Despite these seemingly unsurmountable problems, there is some effort underway to offer an advantage of this sort to compute bound programs. An entirely new and highly parallel architecture is being developed by a company called Tera [99]. They claim to have achieved zero-overhead context switching. To quote from their web page, "Each processor switches tasks after every instruction, with no overhead. By rapidly switching tasks, each processor is able to keep busy, regardless of memory latency or synchronization delays." This is achieved by its multi-threaded architecture (MTA) which includes "virtual processors." Each processor in the Tera machine contains 128 virtual processors, each with its own program counter, register set, stream status word, and target and trap registers. With the zero-overhead context switch, combined with enough threads running on a given processor memory latency can be hidden.

Consider the previously mentioned example. If there are 20 threads running on the virtual processors of a given processor in the MTA, and threads are switched

every instruction, when one thread does a memory access, it has to wait about 20 clock cycles to receive data from main memory. These 20 cycles would then be used by other virtual processors (or threads) thereby using the time for perhaps useful computations.

This new and developing architecture holds great promise for computationally intensive applications [100–103]. Not only is the Tera MTA designed to solve the memory latency problem, but it is also designed to facilitate the development of parallel code because of its shared memory architecture and heavy reliance on threads. We discuss these issues as they relate to molecular dynamics in the next sections.

5.2.2. Parallelization

Most investigators who use molecular dynamics have their own codes that are specifically tailored to their particular needs. Many of these could benefit from conversion to parallel equivalents. However, parallelizing serial codes for use on distributed memory parallel computers is notoriously difficult requiring substantial changes to the architecture of the codes, additional code to handle message passing, and difficult debugging. Multi-threading serial code, on the other hand, is much less painful. Changes to the original code are minimal, and the amount of additional code needed is also very small. The key advantage is that all the memory is shared, hence there is no need to worry about message passing between threads. Of course there are some issues that one needs to be careful of when working with data that may be modified by more than one thread simultaneously, but there are standard routines such as mutex locks and semaphores that are designed to handle these problems.

In addition, there are next-generation compilers that will do automatic parallelization using threads. The Tera corporation is developing compilers for just this purpose.

5.2.3. Easy Load Balancing

Consider running a molecular dynamics simulation on a shared memory multiprocessor machine using threads. It is the responsibility of the operating system to determine on which processor each thread should run, and that of the hardware to keep their caches coherent and up to date. Also, if there are more threads than there are processors, then when a processor becomes free (perhaps because it has finished its work) another thread can be started on that processor to take advantage of the free CPU cycles.

A typical parallelization scheme for molecular dynamics is spatial decomposition [13]. The atoms in a simulation are distributed to processors (or in this case, threads) according to the region of space that they occupy. If atoms are more densely packed in one region of space than another, the thread for that region has more work to do than others. The result is an imbalance in load. Other threads will have to wait at the end of each time-step for the overloaded one to finish.

If there are more threads than available processors, the processors with the threads that finish first will have additional threads to work on. When the number of threads becomes large enough, we should expect to see a load-balancing effect.

For example, if we have four threads, and four processors are available, one thread will run on each processor. At the end of each time-step the thread that finishes earliest will have to wait for all the others to finish. If we have eight threads instead, and the same four processors, each of four threads are assigned to the four

processors, and four will be waiting. When the processor with the fastest thread completes its work, it starts one of the four waiting threads and begins working on it. The other three processors do the same thing. Even in this scenario, some processors may finish (i.e. run out of waiting threads) well before others, but we can increase the number of threads far beyond the number of processors. In the limit of very large numbers of threads, the load is balanced as well as it can be. This fact is offset by the cost of the additional context switching needed to manage all of the threads. Hence, one expects to see an optimal number of threads for a given simulation.

5.3. Implementation

In order to test the viability of these proposed benefits of multi-threading, we conducted tests using a simple liquid argon simulation with a Lennard-Jones inter-atomic potential,

$$\phi(r_{ij}) = \epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] . \tag{5.1}$$

The values of the constants for argon are $\epsilon = 4.156$ kJ/mol and $\sigma = 3.446$ Å.

The simulation is based on the linked–list cell method. We mapped the cells onto a regular mesh throughout the 3-D space of the simulation. There was an average of four atoms per cell.

To distribute the cells to the threads, we defined a "grid" of threads by stating the number of threads in each dimension. For example, if the user wants 27 threads she might define a thread grid with three threads in each dimension, or alternatively, nine in the x dimension, three in the y dimension, and one in the z dimension. The cells are then divided among the threads by dividing the number of

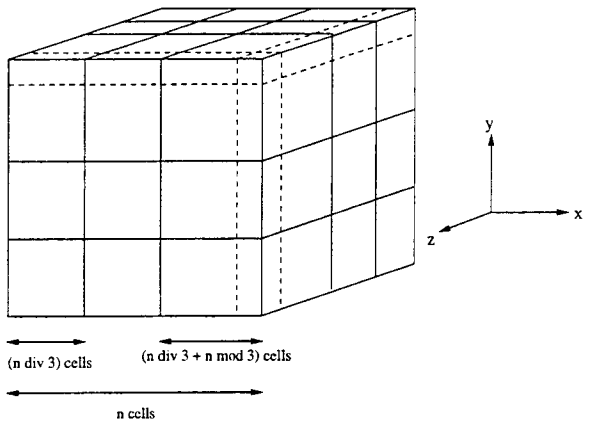

FIGURE 5.1. A $3 \times 3 \times 3$ "grid" of threads.

cells in each dimension by the number of threads in that dimension. The remaining cells, if any, were given to the last thread in each dimension. Hence if there were 27 threads in a $3 \times 3 \times 3$ grid, and there were 64000 cells in a $40 \times 40 \times 40$ grid, then most threads would work on a $13 \times 13 \times 13$ cell area (2197 cells), and the third thread in each dimension would work on a $14 \times 14 \times 14$ or $14 \times 13 \times 13$ or $14 \times 14 \times 13$ cell area. As is shown in Figure 5.1, each thread works on a given area possibly including the small extra part at the far end of each dimension.

We define this "grid" of threads at runtime. The program divides the cells among the threads using the method I have just described. It is obvious that if $n \bmod p = 0$ (p being the number of threads and n being the number of cells in a given dimension), then each thread has the same number of cells to work on. Since the argon simulation is fairly homogeneous, if one thread has more cells than another then it certainly has more work to do, and hence a load imbalance exists among the threads.

In using threads in simulations, one must consider synchronization and memory protection. Whenever an area of memory that can be modified by multiple concurrent threads is changed, it must be protected so that while the given thread is making the change, no other threads may modify it until the change is completely finished. The reason for this is that the way in which load and store instructions are interleaved by the processor is not guaranteed.

This problem can be demonstrated with a simple example. Suppose two threads want to increment the same counter. Typically, in order to increment a variable, a processor will first load the value of the variable into the processor's register, increment it, and then store the new value into the original memory location. Now, if two threads were trying to increment that same variable, the following sequence is possible: Thread 1 loads the value into the register, then thread 1 is pre-empted by thread 2. Thread 2 reads the same value into its register. Now each thread is incrementing the same number, and when they both store their information to the original memory location, the final value of the variable will have only been incremented once instead of twice as intended. The lesson here is to always protect any data that can be modified by multiple threads.

We can protect shared data by using certain routines provided by the thread libraries. The one most commonly used is called a mutex lock (mutual exclusion lock). Shared data is protected by surrounding the critical sections of code with calls to the same mutex lock. A particular thread must first "obtain" the lock before it is allowed to modify the data. An example of this is shown if Figure 5.2, where the variable `m` refers to the mutex lock. When one thread obtains a mutex lock, no other threads can obtain the same lock until the original thread releases the lock by calling `mutex_unlock`.

```
Thread 1                    Thread 2
----------------            -------------------
mutex_lock(&m);             mutex_lock(&m);
a++;                        a = a + 7;
mutex_unlock(&m);           mutex_unlock(&m);
```

FIGURE 5.2. Use of `mutex_lock`.

During the course of a molecular dynamics simulation, a given thread must read the coordinates of all of the particles within its allotment of cells and must also read some of the coordinates of particles in neighboring threads cells. However, unless we take advantage of Newton's third law ($\mathbf{f}_{ij} = -\mathbf{f}_{ji}$) it is not necessary for a given thread to *write* to atoms owned by other threads. Therefore, there is very little need to worry about protecting data! In fact, as it turns out the only places we need to protect are when moving atoms between cells and when computing global sums like total energy and temperature.

However, there is also a need to synchronize threads at the end of each routine to avoid erroneous conditions in which, for example, one thread continues to move its atoms around while another thread is still calculating forces. This is avoided by inserting barrier synchronizations after each step within a time-step. A barrier simply provides a point in the code beyond which no thread can continue until all other threads have reached the same point.

5.4. Results

To investigate the effects of multi-threading, the argon simulation was run using various numbers of threads on various problem sizes. We present here a summary of the results of these test cases.

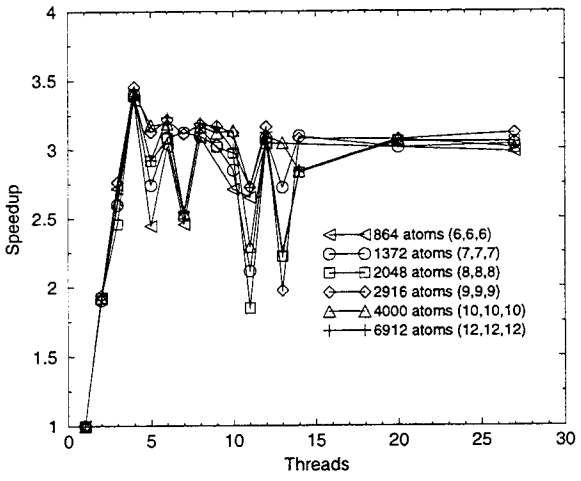

FIGURE 5.3. Speedup for simulations of various sizes on the four processor SPARC-Server 20/51. The numbers in parenthesis (e.g. (10,10,10)) indicate the number of cells in each dimension. There are initially 4 atoms per cell.

First, we compared threaded simulations on single processor machines of various kinds. The execution time for running on a single thread was always faster than the time to execute the same simulation on multiple threads. This confirms the suspicion we stated earlier, namely that threading may not be effective in exploiting the memory bus bottleneck, at least for this particular architecture. Again, we emphasize that this problem cannot be solved without fundamental changes in the architecture.

Next, we consider parallelization and load balancing effects. The test simulation was run on a four processor SPARCServer 20/51. As shown in Figure 5.3, the maximum speedup occurs when there is the same number of threads as processors. As the number of threads is increased beyond the number of processors, the speedup stays slightly below the maximum speedup and gradually declines. Again, this confirms that for current architectures, any advantage from multi-threading on

single processors in a multi-processor system derives from load balancing and cache effects and not on improved performance on the individual processors.

Figure 5.3 also shows abrupt dips in the speedup for numbers of threads that are prime (5,7,11, and 13 threads). This is directly related to the difficulty in dividing the work among threads. If the number of threads is prime, then a cubic region can only divide itself among them in a single dimension. Unless the number of cells in that dimension is divisible by this prime number of threads, one thread will have less work to do than all the others. In addition, since the number of threads is prime, it is impossible to divide them evenly among the four processors. Note also that the work imbalance issue is also a factor whenever the number of cells in a given dimension is not divisible by the number of threads in that dimension.

There is no way for a thread manager to solve the imbalance caused by having a number of equally balanced threads which is not divisible by the number of processors. However, if there is an imbalance in work among the threads, the thread scheduler should be able to help distribute threads among processors as they become available. One would expect also that the larger the number of threads, the better the load imbalance would be managed. This effect can be seen in Figure 5.3. Consider the case of 20 threads. The threads were divided in each dimension as (5,4,1). The number of cells in the first dimension is not evenly divisible by five for all but the 4000 atom case. Hence, a load imbalance exists for most of the different size runs. There is very little spread in the results for each run for the 20 thread case indicating that the load imbalance is handled well by the thread scheduler.

5.5. Concluding Remarks

Our studies indicate that molecular dynamics can benefit from parallelizing by multi-threading. The key advantages are that coding is much simpler and straightforward, thanks to the shared memory architecture, and that load balancing can be done automatically by the thread scheduler. There are also indications that compute bound programs like molecular dynamics can benefit from new architectures currently in development (such as the Tera MTA) which help multi-threaded programs handle memory latencies more efficiently.

An additional benefit to multi-threaded code which we have not mentioned until now, is that the code is portable. All operating systems that comply with the POSIX standard will be able to compile and run multi-threaded programs with almost no changes. This is generally not the case for distributed memory code.

Overall, our studies indicate that multi-threaded molecular dynamics simulations do not have better single-processor performance than their distributed memory counterparts. However, the benefits from ease of code development, portability, and load balancing as well as hardware costs must be weighed to determine which environment is best for a particular simulation.

6. SUMMARY AND CONCLUSIONS

In this thesis we have presented studies that are intended to help the computational science community a step further towards the goal of simulating larger atomic systems more accurately.

We have provided a comparative study of SiO_2 potential energy functions which will enable researchers to determine which interaction gives the most accurate results with respect to experimental data.

One of the most important goals of MD research is to improve our ability to simulate larger systems for longer periods of simulated time. Towards that end, we have provided a study into the use of tabulated potential energy and force functions to accelerate MD calculations. We also investigated the use of multi-threaded programming to parallelize MD codes, to help with load balancing, and to overcome memory latencies.

This thesis has provided investigations that will assist future researchers in developing better potential energy functions for SiO_2 and in optimizing their current molecular dynamics codes for speed and parallelization. We have also discussed the current state of molecular dynamics including algorithms, techniques, tools, and computer architectures.

BIBLIOGRAPHY

[1] M. P. Allen and J. P. Tildesley, *Computer Simulation of Liquids*, Oxford Science Publications, Oxford, 1987.

[2] D. C. Rapaport, *The Art of Molecular Dynamics Simulation*, Cambridge University Press, Cambridge, 1995.

[3] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, Adam Hilger, Bristol, 1988.

[4] K. Refson, *Moldy Users Manual*, For more information see the following web page: `http://www.earth.ox.ac.uk/~keith/moldy.html`.

[5] M. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L. Kale, R. D. Skeel, and K. Schulten, NAMD – A parallel, object–oriented molecular dynamics program, Journal of Supercomputing Applications and High Performance Computing (1996).

[6] P. Clapp and J. Rifkin, Personal communication. See the following web page: `http://www.ims.uconn.edu/centers/simul/`.

[7] D. Turner and J. Morris, The ALCMD molecular dynamics code. In conjunction with Ames Laboratory and U.S. Department of Energy, See: `http://cmp.ameslab.gov/cmp/CMP_Theory/cmd/alcmd_source.html`.

[8] D. M. Beazley and P. S. Lomdahl, Message–passing multi–cell molecular dynamics on the Connection Machine 5, Parall. Comp. **20**, 173 (1994).

[9] A. Nakano, R. K. Kalia, and P. Vashishta, Dynamics and morphology of brittle cracks: A molecular–dynamics study of silicon nitride, Phys. Rev. Lett. **75**, 3138 (1995).

[10] W. Li, R. K. Kalia, and P. Vashishta, Amorphization and fracture in silicon diselenide nanowires: A molecular dynamics study, Phys. Rev. Lett. **77**, 2241 (1996).

[11] S. J. Zhou, D. M. Beazley, P. S. Lomdahl, and B. L. Holian, Large–scale molecular dynamics simulations of three–dimensional ductile failure, Phys. Rev. Lett. **78**, 479 (1997).

[12] S. Gupta, Computing aspects of molecular dynamics simulation, Computer Physics Communications **70**, 243 (1992).

[13] S. Plimpton, Fast parallel algorithms for short–range molecular dynamics, Journal of Computational Physics **117**, 1 (1995).

[14] S. Plimpton and B. Hendrickson, A new parallel method for molecular dynamics simulation of macromolecular systems, J. of Computational Chemistry **17**, 326 (1996).

[15] P. Vashishta, R. K. Kalia, S. W. de Leeuw, D. L. Greenwell, A. Nakano, W. Jin, J. Yu, L. Bi, and W. Li, Computer simulation of materials using parallel architectures, Computational Materials Science **2**, 180 (1994).

[16] A. Nakano, P. Vashishta, and R. K. Kalia, Parallel multiple–time–step molecular dynamics with three–body interaction, Computer Physics Communications **77**, 303 (1993).

[17] P. S. Lomdahl and D. M. Beazley, Multi–million particle molecular dynamics on MPPs, in *Lecture notes in computer science*, volume 1041, page 391, Springer–Verlag, 1996.

[18] W. Smith, Molecular dynamics on hypercube parallel computers, Computer Physics Communications **62**, 229 (1991).

[19] D. Fincham, Parallel computers and molecular simulation, Molecular Simulation **1**, 1 (1987).

[20] Y.-S. Hwand, R. Das, J. H. Saltz, M. Hodošček, and B. R. Brooks, Parallel molecular dynamics programs for distributed–memory machines, IEEE Computational Science and Engineering , 18 (summer 1995).

[21] Y. Deng, A. McCoy, R. B. Marr, R. F. Peierls, and O. Yaşar, Molecular dynamics for 400 million particles with short-range interactions, in *High Performance Computing Symposium 1995, Grand Callenges in Computer Simulation, Proceedings of the 1995 Simulation Multiconference, April 9–13, 1995, Phoenix, AZ*, edited by A. Tentner, page 95, 1995.

[22] M. Dormanns and W. Sprangers, Experiences with asynchronous parallel molecular dynamics simulations, in *Proceedings High Performance Computing and Networking Europe '96*, page 213, Springer Publ., 1996.

[23] M. Dormanns, W. Sprangers, and T. Bemmerl, Feasibility of asynchronous parallel molecular dynamics simulations, in *High Performance Computing Conference (HPC) '96*, page 23, 1996.

[24] C.-E. Hong and B. M. McMillin, Relaxing synchronization in distributed simulated annealing, IEEE Transactions on Parallel and Distributed Systems **6**, 189 (1995).

[25] M. Dormanns and W. Sprangers, On the precision of asynchronous paralle molecular dynamics simulations, Technical report, RWTH Aachen, Lehrstuhl für Betriebssysteme, 1995.

[26] F. F. Abraham, J. Q. Broughton, N. Bernstein, and E. Kaxiras, Spanning the length scales in dynamic simulation, Computers in Physics **12**, 538 (1998).

[27] D. M. Beazley and P. S. Lomdahl, Controlling the data glut in large–scale molecular–dynamics simulations, Computers in Physics **11**, 230 (1997).

[28] D. M. Beazley and P. S. Lomdahl, Lightweight computational steering of very large molecular dynamics simulations, in *Supercomputing '96*, 1996.

[29] D. M. Beazley and P. S. Lomdahl, Extensible message passing application development and debugging with python, in *IPPS '97*, 1997.

[30] D. M. Beazley and P. S. Lomdahl, Building flexible large–scale scientific computing applications with scripting languages, in *8th SIAM Conference on Parallel Processing for Scientific Computing*, 1997.

[31] A. Nakano, R. K. Kalia, and P. Vashishta, First sharp diffraction peak and intermediate–range order in amorphous silica: Finite–size effects in molecular dynamics simulations, Journal of Non–Crystalline Solids **171**, 157 (1994).

[32] F. H. Stillinger and T. A. Weber, Computer simulation of local order in condensed phases of silicon, Phys. Rev. B **31**, 5262 (1985).

[33] W. Xu and J. A. Moriarty, Atomistic simulation of ideal shear strength, point defects, and screw dislocations in bcc transition metals: Mo as a prototype, Phys. Rev. B **54**, 6941 (1996).

[34] T. L. Hill, *An introduction to statistical thermodynamics*, Dover, 1986.

[35] C. W. Gear, *Numerical initial value problems in ordinary differential equations*, Prentice–Hall, Englewood Cliffs, NJ, 1971.

[36] P. Vashishta, R. K. Kalia, A. Nakano, W. Li, and I. Ebbsjö, Molecular dynamics methods and large–scale simulations of amorphous materials, in *Amorphous Insulators and Semiconductors*, edited by M. F. Thorpe and M. I. Mitkova, page 151, Kluwer Academic Publishers, Netherlands, 1997.

[37] A. Y. Toukmaji and J. John A. Board, Ewald summation techniques in perspective: A survey, Computer Physics Communications **95**, 73 (1996).

[38] C. G. Lambert, T. A. Darden, and J. A. B. Jr., A multipole–based algorithm for efficient calculation of forces and potentials in macroscopic periodic assemblies of particles, J. of Computational Physics **126**, 274 (1996).

[39] C. A. White and M. Head-Gordon, Derivation and efficient implementation of the fast multipole method, J. Chem. Phys. **101**, 6593 (1994).

[40] T. Hrycak and V. Rokhlin, An improved fast multipole algorithm for potential fields, SIAM J. Sci. Comput. **19**, 1804 (1998).

[41] W. D. Elliott and J. A. B. Jr., Fast fourier transform accelerated fast multipole algorithm, SIAM J. Sci. Comput. **17**, 398 (1996).

[42] D. Fincham, Optimization of the Ewald sum for large systems, Molecular Simulation **13**, 1 (1994).

[43] C. K. Loong, P. Vashishta, R. K. Kalia, W. Jin, M. H. Degani, D. G. Hinks, D. L. Price, J. D. Jorgensen, B. Dabrowski, A. W. Mitchell, D. R. Richards, and Y. Zheng, Phonon density of states and oxygen–istotope effect in $Ba_{1-x}K_xBiO_3$, Phys. Rev. B **45**, 8052 (1992).

[44] T. F. Soules, Molecular dynamic calculations of glass structure and diffusion in glass, Journal of Non–Crystalline Solids **49**, 29 (1982).

[45] T. F. Soules, Computer simulation of glass structures, Journal of Non–Crystalline Solids **123**, 48 (1990).

[46] W. Kob, Computer simulations of supercooled liquids and glasses, To appear as a Topical Review Article in J. Phys: Condens. Matter.

[47] R. J. Bell and P. Dean, The Structure of vitreous silica: Validity of the random network theory, Philos. Mag. **25**, 1381 (1972).

[48] A. C. Wright, Neutron scattering from vitreous silica. V. The structure of vitreous silica: What have we learned from 60 years of diffraction studies?, Journal of Non–Crystalline Solids **179**, 84 (1994).

[49] R. N. Sinclair and A. C. Wright, Neutron scattering from vitreous silica. I. The total cross–section, Journal of Non–Crystalline Solids **57**, 447 (1983).

[50] P. A. V. Johnson, A. C. Wright, and R. N. Sinclair, Neutron scattering from vitreous silica. II. Twin–axis diffraction experiments, Journal of Non–Crystalline Solids **58**, 109 (1983).

[51] A. C. Wright and R. N. Sinclair, Neutron scattering from vitreous silica. III. Elastic diffraction, Journal of Non–Crystalline Solids **76**, 351 (1985).

[52] D. I. Grimley, A. C. Wright, and R. N. Sinclair, Neutron scattering from vitreous silica. IV. Time–of–flight diffraction, Journal of Non–Crystalline Solids **119**, 49 (1990).

[53] S. Susman, K. J. Volin, D. G. Montague, and D. L. Price, Temperature dependence of the first sharp diffraction peak in vitreous silica, Phys. Rev. B **43**, 11076 (1991).

[54] S. Susman, K. J. Volin, D. L. Price, M. Grimsditch, J. P. Rino, R. K. Kalia, P. Vashishta, G. Gwanmesia, Y. Yang, and R. C. Liebermann, Intermediate–range order in permanently densified vitreous SiO_2: A neutron–diffraction and molecular–dynamics study, Phys. Rev. B **43**, 1194 (1991).

[55] S. N. Taraskin and S. R. Elliott, Nature of vibrational excitations in vitreous silica, Phys. Rev. B **56**, 8605 (1997).

[56] S. H. Garofalini, Molecular dynamics simulation of the frequency spectrum of amorphous silica, J. Chem. Phys. **76**, 3189 (1982).

[57] B. Guillot and Y. Guissani, Boson peak and high frequency modes in amorphous silica, Phys. Rev. Lett. **78**, 2401 (1997).

[58] J. Horbach, W. Kob, and K. Binder, The dynamics of supercooled silica: Acoustic modes and boson peak, Journal of Non–Crystalline Solids **235**, 237 (1998).

[59] M. Wilson, P. A. Madden, M. Hemmati, and C. A. Angell, Polarization effects, network dynamics, and the infrared spectrum of amorphous SiO_2, Phys. Rev. Lett. **77**, 4023 (1996).

[60] M. Hemmati and C. A. Angell, IR absorption of silicate glasses studied by ion dynamics computer simulation. I. IR spectra of SiO_2 glass in the rigid ion model approximation, Journal of Non–Crystalline Solids **217**, 236 (1997).

[61] D. C. Anderson, J. Kieffer, and S. Klarsfeld, Molecular dynamics simulations of the infrared dielectric response of silica structures, J. Chem. Phys. **98**, 8978 (1993).

[62] R. Car and M. Parrinello, Unified Approach for molecular dynamics and density-functional theory, Phys. Rev. Lett. **55**, 2471 (1985).

[63] A. A. Demkov, J. Ortega, O. F. Sankey, and M. P. Grumbach, Electronic structure approach for complex silicas, Phys. Rev. B **52**, 1618 (1995).

[64] J. Sarnthein, A. Pasquarello, and R. Car, Structural and electronic properties of liquid and amorphous SiO_2: An *ab initio* molecular dynamics study, Phys. Rev. Lett. **74**, 4682 (1995).

[65] J. Sarnthein, A. Pasquarello, and R. Car, Model of vitreous SiO_2 generated by an *ab initio* molecular dynamics quench from the melt, Phys. Rev. B **52**, 12690 (1995).

[66] A. Pasquarello, J. Sarnthein, and R. Car, Dynamic structure factor of vitreous silica from first principles: Comparison to neutron–inelastic–scattering experiments, Phys. Rev. B **57**, 14133 (1998).

[67] J. Horbach, W. Kob, and C. A. Angell, Finite size effects in simulations of glass dynamics, Phys. Rev. E **54**, 5897 (1996).

[68] J. Horbach, W. Kob, and K. Binder, Molecular dynamics computer simulation of the dynamics of supercooled silica, Philos. Mag. B **77**, 297 (1998).

[69] W. Kob, J. Horbach, and K. Binder, The dynamics of non–crystalline silica: Insight from molecular dynamics computer simulations, To be published.

[70] A. Nakano, R. K. Kalia, and P. Vashishta, Growth of pore interfaces and roughness of fracture surfaces in porous silica: Million particle molecular–dynamics simulations, Phys. Rev. Lett. **73**, 2336 (1994).

[71] P. Vashishta, R. K. Kalia, A. Nakano, and W. Jin, Silica under very large positive and negative pressures: Molecular dynamics simulations on parallel computers, International Journal of Thermophysics **17**, 169 (1996).

[72] M. Hemmati and C. A. Angell, Comparison of pair potential models for the simulation of liquid SiO_2: Thermodynamic, angular distribution, and diffusional properties, in *Physics Meets Geology*, Cambridge Univ. Press, Cambridge, UK, 1998, (in press).

[73] B. Vessal, M. Amini, D. Fincham, and C. R. A. Catlow, Water–like melting behaviour of SiO_2 investigated by the molecular dynamics simulation technique, Philosophical Magazine B **60**, 753 (1989).

[74] B. W. H. van Beest, G. J. Kramer, and R. A. van Santen, Force fields for silicas and aluminophosphates based on *ab initio* calculations, Phys. Rev. Lett. **64**, 1955 (1990).

[75] S. Tsuneyuki, M. Tsukada, H. Aoki, and Y. Matsui, First–principles interatomic potential of silica applied to molecular dynamics, Phys. Rev. Lett. **61**, 869 (1988).

[76] P. Vashishta, R. K. Kalia, J. P. Rino, and I. Ebbsjö, Interaction potential for SiO_2: A molecular–dynamics study of structural correlations, Phys. Rev. B **41**, 12197 (1990).

[77] S. Tsuneyuki, H. Aoki, M. Tsukada, and Y. Matsui, Molecular–dyanmics study of the α to β structural phase transition of quartz, Phys. Rev. Lett. **64**, 776 (1990).

[78] J. R. Rustad, D. A. Yuen, and F. J. Spera, Molecular dynamics of liquid SiO_2 under high pressure, Phys. Rev. A **42**, 2081 (1990).

[79] K. Vollmayr, W. Kob, and K. Binder, Cooling–rate effects in amorphous silica: A computer–simulation study, Phys. Rev. B **54**, 15808 (1996).

[80] R. L. Mozzi and B. E. Warren, The structure of vitreous silica, J. Appl. Cryst. **2**, 164 (1969).

[81] J. H. Konnert and J. Karle, The computation of radial distribution functions for glassy materials, Acta Cryst. A **29**, 702 (1973).

[82] R. F. Pettifer, R. Dupree, I. Farnan, and U. Sternberg, NMR determinations of Si–O–Si bond angle distributions in silica, Journal of Non–Crystalline Solids **106**, 408 (1988).

[83] P. G. Coombs, J. F. DeNatale, P. J. Hood, D. K. McElfresh, R. S. Wortman, and J. F. Shackelford, The nature of the Si–O–Si bond angle distribution in vitreous silica, Philos. Mag. B **51**, L39 (1985).

[84] J. R. G. DaSilva, D. G. Pinatti, C. E. Anderson, and M. L. Rudee, A refinement of the structure of vitreous silica, Philos. Mag. **31**, 713 (1975).

[85] C. A. Angell and L. M. Torell, Short time structural relaxation processes in liquids: Comparison of experimental and computer simulation glass transitions on picosecond time scales, J. Chem. Phys. **78**, 937 (1983).

[86] C. A. Angell, J. H. R. Clarke, and L. V. Woodcock, Interaction potentials and glass formation: A survey of computer experiments, Adv. Chem. Phys. **48**, 397 (1981).

[87] L. V. Woodcock, C. A. Angell, and P. Cheeseman, Molecular dynamics studies of the vitreous state: Simple ionic systems and silica, J. Chem. Phys. **65**, 1565 (1976).

[88] C. A. Angell, Dynamic processes in ionic glasses, Chem. Rev. **90**, 523 (1990).

[89] S. R. Elliot, *Physics of Amorphous Materials*, Longman Group Limited, England, 1983.

[90] M. Arai, A. C. Hannon, A. D. Taylor, T. Otomo, A. C. Wright, R. N. Sinclair, and D. L. Price, Neutron scattering law measurements for vitreous silica, Trans. Am. Crystallog. Assoc. **27**, 113 (1991).

[91] D. L. Price and J. M. Carpenter, Scattering function of vitreous silica, Journal of Non–Crystalline Solids **92**, 153 (1987).

[92] P. Vashishta and R. Kalia, 1997, Private communications.

[93] D. E. Sanders, M. S. Stave, L. S. Perkins, and A. E. DePristo, SCT89: A computer code for atomic and molecular scattering from clean and adsorbate covered surfaces, Comp. Phys. Commun. **70**, 579 (1992).

[94] M. S. Stave, D. E. Sanders, T. J. Raeker, and A. E. DePristo, Corrected effective medium method. V. Simplifications for molecular dynamics and Monte Carlo simulations, J. Chem. Phys. **93**, 4413 (1990).

[95] T. A. Andrea, W. C. Swope, and H. C. Andersen, The role of long-ranged forces in determining the structure and properties of liquid water, Journal of Chemical Physics **79**, 4576 (1983).

[96] B. Lewis and D. J. Berg, *Threads Primer: A Guide to Multithreaded Programming*, SunSoft Press, 1996.

[97] S. Kleiman, D. Shah, and B. Smaalders, *Programming with Threads*, SunSoft Press, 1996.

[98] R. Stevens, 1997, Private communications.

[99] `http://www.tera.com`.

[100] A. Snavely, G. Johnson, and J. Genetti, Data intensive volume visualization on the Tera MTA and Cray T3E, in *ASTC '99 Proceedings*, 1999.

[101] A. Snavely, L. Carter, J. Boisseau, A. Majumdar, K. S. Gatlin, N. Mitchell, J. Feo, and B. Koblenz, Multi-processor performance on the Tera MTA, in *SC '98 Proceedings*, 1998.

[102] S. H. Bokhari and D. J. Mavriplis, The Tera multithreaded architecture and unstructured meshes, Technical Report NASA/CR-1998-208953, Langley Research Center, 1998, ICASE Interim Report No. 33.

[103] A. Snavely, N. Mitchell, L. Carter, J. Ferrante, and D. Tullsen, Explorations in symbiosis on two multithreaded architectures, in *M-TEAC '99 Proceedings*, 1999.