

# **Toolkit-based framework for scalable High Performance Standalone Molecular Dynamics simulations**



**Richard Opio Ocaya**

**(Promoter: Professor J.J. Terblans)**

Department of Physics, Faculty of Natural & Agricultural Sciences  
University of the Free State

A thesis submitted in fulfilment of the requirements for the degree of  
*Philosophiae Doctor*

Student number: 2014216382

April 2019

In memory of my mother ...

## Declaration

I hereby declare that, except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Richard Opio Ocaya

---

*Signature*

---

*Date*

## **Acknowledgements**

I wish to acknowledge with gratitude my Promoter, Professor J.J. Terblans, for his contributions and guidance.

I am also grateful to my colleagues in the Department of Physics for their camaraderie over the years that has created an atmosphere in the Physics Department at the Qwaqwa Campus of the University of the Free State that is conducive for sustained teaching and research, in spite of the many, enduring challenges.

I thank my children for enduring this journey with me with fortitude.  
Blessed be the Name of the Lord.

## **Abstract**

Computational modelling and simulation in materials science uses mathematical abstractions of particle-particle forces to postulate, develop and understand materials that are organized as particle systems. Real particle systems occupy macroscopic scales and can be costly to simulate in terms of hardware and software tools and simulation time. Even the most basic simulation can generate large amounts of intermediate data that requires innovative further processing to decipher its underlying physics or to answer fundamental questions about its material properties. These questions are increasingly being asked due the present furious academic and industrial interest in nanosized crystalline lattices. Of particular pertinence are questions of whether or not the properties of the nanostructures are identical to those of their macrostructures. In the light of this the primary focus of this contribution is the development of a tool to simulate face-centered cubic (fcc) particle systems on a “standalone” hardware platform, and to apply it to a specified particle system. The studied particle systems are to range from nanostructures to macrostructures.

This thesis is thematically divided into two main parts. In the first part, comprising the first five chapters, we conduct a detailed survey of the current in computation, followed by a definition of the kinds of systems to which the study is applicable, and then we provide a detailed but not overwhelming description of the tool development, with numerous actual codes and examples. This part culminates in a working tool, abbreviated VSV. In the second part, comprising the subsequent chapters, we apply the VSV and associated tools to solve

actual physics problems in nanostructures thereby offering new approaches and results to answer the current questions.

In developing VSV, we discuss the pairwise-particle potential and its integration into an embedded atom model (EAM) approach that is a cost-effective way to simulate fcc metallic lattice systems as a select case that has practical and industrial relevance. To do this, We chose the Sutton-Chen EAM as being suitable. This was followed by the application of VSV on a single computer as the “standalone” setting, and then on a small, four-computer cluster consisting of multi-core, multi-processors to test its scaling and parallelization. The test system consisted of 30,261 copper atoms in an arbitrary fcc lattice. The various simulations were then evaluated for performance enhancements in terms of execution speed and ease of application. The large amounts of intermediate data made it necessary to develop smaller extensions to the VSV tool to enable output visualization. These extensions were written in Visual Basic and Matlab. We then apply VSV to simulate the systems at low energies and suggest novel answers to various questions within the framework mentioned above.

A first major, unwittingly observed result in the application of VSV is that it showed that bond lengths between any two particles appear to develop a temporal oscillation when perturbed by a nearby displaced atom. These oscillations are seen to propagate throughout the lattice and eventually form a standing wave pattern, through which temperature can be modelled. By applying perturbations in which the bond length oscillation amplitudes are constrained to small values by deliberately applied perturbations, we found that the use of an elastic, Hooke’s Law model results in a faithful reproduction of the known elastic constants for the copper material on which it was tested. Thus, we suggest and develop a unique impulse and oscillation method that is useful to calculate the elastic properties of fcc nanostructures. A second result is the extension of this impulse-oscillation method to postulate a way to initiate wave-like energy transfer through the lattice. These waves are shown to be phonons and by constraining the energy to the first Brillouin zone we show that

the temperature behavior of the lattice can be estimated. A third result is that VSV enables the computation of the diffusion in a nanosized lattice. Furthermore, we apply standard diffusion models to a low temperature regime and calculate the diffusion constants of the lattice using standard models. An important result is the indication that diffusion in such a collection of atoms is not driven by Brownian motion, but by the interplay of pair-wise 12-6 forces. We also show that the lattice atoms can spontaneously coalesce into a shape that is guided by the global minimum of potential energy. Finally, VSV also shows how growth into bigger fcc structures can be simulated through atom captures.

The foregoing results are compared with the literature values and the good agreements indicate that it is a reasonable new and cost effective way to investigate many of the properties of fcc lattices. Aspects of the research have been published in a book chapter, journals and presented at international conferences.

Finally, we present the concluding remarks about research and suggest further directions in terms of further development of VSV and its applications to new and novel structures.

# Table of contents

|  |              |
|--|--------------|
| <b>Abstract</b>  | <b>v</b>     |
| <b>List of figures</b>                                     | <b>xiv</b>   |
| <b>List of tables</b>                                      | <b>xvii</b>  |
| <b>List of C-program codes</b>                             | <b>xviii</b> |
| <b>Nomenclature</b>  | <b>xix</b>   |
| <b>1 General Introduction</b>                              | <b>1</b>     |
| 1.1 Overview . . . . .                                     | 1            |
| 1.1.1 Computational modelling . . . . .                    | 1            |
| 1.1.2 Main approaches in computational modelling . . . . . | 2            |
| 1.1.3 Parallelization . . . . .                            | 3            |
| 1.1.4 Decomposition and task allocation . . . . .          | 4            |
| 1.1.5 Statement of the problem . . . . .                   | 5            |
| 1.1.6 Objectives of the research . . . . .                 | 6            |
| 1.1.7 Outline of the thesis . . . . .                      | 7            |
| <b>References</b>  | <b>11</b>    |

---

|          |   |           |
|----------|---|-----------|
| <b>2</b> | <b>Designing a MD simulation testbed</b>                        | <b>16</b> |
| 2.1      | Introduction . . . . .  | 16        |
| 2.1.1    | Classification of parallelization paradigms . . . . .           | 17        |
| 2.1.2    | Coding versus proprietary software . . . . .                    | 17        |
| 2.2      | General tools . . . . .   | 18        |
| 2.2.1    | Parallelizable tools . . . . .                                  | 19        |
| 2.2.2    | Threads and message passing . . . . .                           | 20        |
| 2.2.3    | Open multiprocessing programming . . . . .                      | 22        |
| 2.2.4    | MPI programming . . . . .                                       | 25        |
| 2.2.5    | GPU computing . . . . .   | 25        |
| 2.2.6    | Cloud virtualization . . . . .                                  | 26        |
| 2.2.7    | Specifications of the standalone testbed . . . . .              | 27        |
| 2.3      | Chapter summary . . . . .                                       | 28        |
|          | <b>References</b>   | <b>29</b> |
| <b>3</b> | <b>Development of the simulation software</b>                   | <b>32</b> |
| 3.1      | Introduction . . . . .  | 32        |
| 3.1.1    | The canonical ensemble . . . . .                                | 32        |
| 3.2      | Defining a particle system . . . . .                            | 33        |
| 3.2.1    | Force in atomic particle systems . . . . .                      | 34        |
| 3.3      | Embedded Atom Modeling . . . . .                                | 34        |
| 3.3.1    | The Finnis-Sinclair approach . . . . .                          | 35        |
| 3.3.2    | The Sutton-Chen form of the Finnis-Sinclair potential . . . . . | 36        |
| 3.3.3    | MD simulation using adatoms . . . . .                           | 37        |
| 3.4      | Velocity Störmer-Verlet integration . . . . .                   | 38        |
| 3.4.1    | Determination of phase space . . . . .                          | 38        |

---

|          |  |           |
|----------|--|-----------|
| 3.5      | Dimensionless equations . . . . .                          | 39        |
| 3.5.1    | Time . . . . .   | 40        |
| 3.5.2    | Velocity and acceleration . . . . .                        | 40        |
| 3.5.3    | Force and pressure . . . . .                               | 41        |
| 3.5.4    | Kinetic and potential energy . . . . .                     | 41        |
| 3.5.5    | Temperature . . . . .                                      | 41        |
| 3.5.6    | Instantaneous and internal pressure . . . . .              | 41        |
| 3.6      | Setting simulation time scale . . . . .                    | 42        |
| 3.7      | Evaluation of diffusion . . . . .                          | 43        |
| 3.8      | Specifying the particle array . . . . .                    | 44        |
| 3.8.1    | Software functionalities . . . . .                         | 48        |
| 3.8.2    | Main code snippets . . . . .                               | 54        |
|          | <b>References</b>  | <b>55</b> |
| <b>4</b> | <b>Performance of the VSV software</b>                     | <b>56</b> |
| 4.1      | Introduction . . . . .                                     | 56        |
| 4.2      | Timing functions . . . . .                                 | 57        |
| 4.3      | Results of energetics and thermostat comparisons . . . . . | 59        |
| 4.4      | Timing performance . . . . .                               | 60        |
| 4.5      | Investigation of real clusters . . . . .                   | 61        |
|          | <b>References</b>  | <b>62</b> |
| <b>5</b> | <b>Specifying system ergodicity in VSV</b>                 | <b>64</b> |
| 5.1      | Introduction . . . . .                                     | 64        |
| 5.2      | Thermostat definitions . . . . .                           | 65        |
| 5.2.1    | Energy equipartition methods . . . . .                     | 65        |

|  |  |                |
|--|--|----------------|
| 5.2.2  | Langevin EPT methods . . . . .                                 | 67             |
| 5.2.3  | Velocity-scaled EPT thermostats . . . . .                      | 67             |
| 5.2.4  | Monte Carlo methods . . . . .                                  | 70             |
| 5.2.5  | Timescale and macroscopics . . . . .                           | 71             |
| 5.3  | Comparative simulation of EPT and MC thermostats . . . . .     | 72             |
| <br><b>References</b>  |  | <br><b>75</b>  |
| <br><b>6 Low temperature diffusion and coalescence using VSV</b>       |  | <br><b>80</b>  |
| 6.1  | Introduction . . . . .   | 80             |
| 6.2  | The simulation model . . . . .                                 | 82             |
| 6.3  | Calculations . . . . .   | 83             |
| 6.3.1  | Determination of cluster-adatom interaction distance . . . . . | 83             |
| 6.3.2  | Single adatom diffusion . . . . .                              | 83             |
| 6.3.3  | Multiple adatom diffusion . . . . .                            | 86             |
| 6.3.4  | Projected adatoms . . . . .                                    | 89             |
| 6.3.4.1  | Projected single adatoms . . . . .                             | 89             |
| 6.3.4.2  | Projected multi-adatoms . . . . .                              | 92             |
| 6.3.5  | Simulating lattice assimilation growth . . . . .               | 94             |
| 6.3.6  | Surface diffusion . . . . .                                    | 96             |
| 6.4  | Conclusions . . . . .  | 100            |
| <br><b>References</b>  |  | <br><b>101</b> |
| <br><b>7 Detection of lattice phonons and their propagation in VSV</b> |  | <br><b>108</b> |
| 7.1  | Introduction . . . . .   | 108            |
| 7.1.1  | Impulse-oscillation approach . . . . .                         | 110            |
| 7.2  | The simulation model . . . . .                                 | 113            |

---

|  |  |            |
|--|--|------------|
| 7.2.1  | Simulation conditions and delimitation . . . . .     | 113        |
| 7.2.2  | Results and discussions . . . . .                    | 116        |
| 7.2.3  | Bond-length oscillations . . . . .                   | 116        |
| 7.2.4  | Wave propagation and the elastic constants . . . . . | 121        |
| 7.2.4.1  | Estimating C11 and C44 . . . . .                     | 124        |
| 7.2.4.2  | Expected temperature rise . . . . .                  | 126        |
| 7.3  | Conclusions . . . . .                                | 126        |
| <b>References</b>                                |  | <b>128</b> |
| <b>8 Post-MD simulation visualization in VSV</b> |  | <b>131</b> |
| 8.1  | Introduction . . . . .                               | 131        |
| 8.2  | Motivation and significance . . . . .                | 132        |
| 8.3  | Description of additional applets . . . . .          | 133        |
| 8.3.1  | Helper routines . . . . .                            | 133        |
| 8.3.1.1  | Highlighting adatoms for VMD . . . . .               | 134        |
| 8.3.1.2  | Tracking atoms on vibrating planes . . . . .         | 136        |
| 8.3.1.3  | Spectral response through bond length . . . . .      | 137        |
| 8.4  | Illustrative example . . . . .                       | 140        |
| 8.4.1  | Elastic constants . . . . .                          | 141        |
| 8.4.2  | Thermography . . . . .                               | 142        |
| 8.5  | Impact . . . . .                                     | 144        |
| 8.6  | Conclusions . . . . .                                | 147        |
| <b>References</b>                                |  | <b>149</b> |
| <b>9 Conclusions</b>                             |  | <b>152</b> |

---

|   |            |
|---|------------|
| <b>10 Publications</b>                                    | <b>156</b> |
| 10.1 Refereed journal articles . . . . .                  | 156        |
| 10.2 Conference presentations . . . . .                   | 157        |
| 10.3 Chapters in books . . . . .                          | 157        |
| 10.4 Software . . . . .                                   | 157        |
| 10.4.1 Permanent VSV Elsevier code repository . . . . .   | 157        |
| 10.4.2 Surface detect applets . . . . .                   | 158        |
| 10.5 Manuscripts under preparation . . . . .              | 158        |
| <br>  |            |
| <b>Appendix A A guide to using VSV for MD simulations</b> | <b>159</b> |
| A.1 Running the VSV code . . . . .                        | 159        |
| A.1.1 Windows 7.0 and later . . . . .                     | 159        |
| A.1.2 The binary executable . . . . .                     | 160        |
| A.1.3 On Windows debug . . . . .                          | 160        |
| A.1.4 Linux e.g. Ubuntu . . . . .                         | 163        |
| A.2 The GNU General Public License . . . . .              | 163        |
| A.3 VSV code listings . . . . .                           | 163        |
| A.3.1 The library file . . . . .                          | 163        |
| A.3.2 The main program . . . . .                          | 176        |
| A.4 Helper applications . . . . .                         | 179        |
| A.4.1 Conversion to VMD format by Excel macro . . . . .   | 179        |
| A.4.2 Spectral response calculation . . . . .             | 183        |
| <br>  |            |
| <b>Appendix References</b>                                | <b>186</b> |

# List of figures

|     |   |    |
|-----|---|----|
| 2.1 | Depiction of the testbed standalone and cluster hardware . . . . .  | 28 |
| 3.1 | The canonical ensemble. . . . .   | 33 |
| 3.2 | Example of an atomic arrangement of copper atoms used in the text. . . . .                                      | 45 |
| 3.3 | VSV program flowchart . . . . .   | 49 |
| 3.4 | The Perturb & Evaluate approach used in VSV simulations . . . . .   | 51 |
| 3.5 | Bird's eye views of simulation fcc lattice . . . . .  | 52 |
| 3.6 | Surface vacancy scheme illustrated using (001) plane . . . . .  | 52 |
| 3.7 | Diagrams showing an atom embedding on (001) plane . . . . .   | 53 |
| 3.8 | Pictorial representation of the VSV program. . . . .  | 54 |
| 4.1 | Speedup measurement approach . . . . .  | 58 |
| 5.1 | Illustration of two possible atom migrations using fcc lattice . . . . .  | 74 |
| 6.1 | Determination of $r_{cut}(\text{min})$ . . . . .  | 84 |
| 6.2 | Final position of captured adatom (red) from $20\text{\AA}$ . The arrow shows the projection direction. . . . . | 85 |

---

|      |   |     |
|------|---|-----|
| 6.3  | Alternate views showing localized environment of the captured adatom (red) that was initially at 4Å above the surface. The reference square in (a) shows the reference positions of the (100) face atoms, (b) is the side view, (c) is the measurement view. The numbers show the same main lattice atoms in the different views. . . . . | 85  |
| 6.4  | Diffusive capture of ten adatoms . . . . .  | 86  |
| 6.5  | Diffusion of 10 adatoms . . . . .   | 87  |
| 6.6  | Coalescing of 10 adatoms . . . . .  | 88  |
| 6.7  | Projected directions . . . . .  | 89  |
| 6.8  | Projection of 10 adatoms . . . . .  | 90  |
| 6.9  | Cohesive energy . . . . .   | 90  |
| 6.10 | Lattice oscillations . . . . .  | 91  |
| 6.11 | Resolved components of radial motion of adatom . . . . .  | 92  |
| 6.12 | Capture of projected adatoms . . . . .  | 93  |
| 6.13 | Distance between adatom and its nearest neighbor over time showing oscillations at the cluster surface. . . . .   | 94  |
| 6.14 | Determination of crystal growth . . . . .   | 95  |
| 6.15 | Near-neighbor distance between the three captured copper atoms. The fourth plot, labeled “REF”, is a reference plot based on the bond length between two randomly selected adjacent Cu atoms that were initially in the main cluster.   | 96  |
| 6.16 | Calculated diffusivity in Cu (N=10) . . . . .   | 98  |
| 6.17 | Region C, showing exponential decay . . . . .   | 99  |
| 7.1  | Mass spring representation of fcc structure . . . . .   | 111 |
| 7.2  | Mass spring representation of fcc structure in the $x$ axis . . . . .   | 112 |
| 7.3  | Passage of impact-generated wave . . . . .  | 115 |
| 7.4  | Example of bond-length versus time . . . . .  | 117 |

---

|      |   |     |
|------|---|-----|
| 7.5  | Simulated bond length amplitude-time and phase-frequency response at the point of impact . . . . .              | 118 |
| 7.6  | Simulated bond length amplitude-time and phase-frequency response along the 100 direction in the bulk . . . . . | 119 |
| 7.7  | Simulated bond length amplitude-time and phase-frequency response along the 110 direction in the bulk . . . . . | 120 |
| 8.1  | Simulated structure of 2281 Cu atoms . . . . .  | 134 |
| 8.2  | GUI of the MS Excel macro . . . . .   | 136 |
| 8.3  | Passage of impact-generated wave . . . . .  | 137 |
| 8.4  | Bond lengths used as illustration . . . . .   | 138 |
| 8.5  | Bond length amplitude-time simulation . . . . .   | 139 |
| 8.6  | FFT of bond lengths . . . . .   | 140 |
| 8.7  | Dispersion . . . . .  | 141 |
| 8.8  | Surface temperature . . . . .   | 142 |
| 8.9  | Thermographic plane . . . . .   | 143 |
| 8.10 | Thermograms after 605.55 fs . . . . .   | 145 |
| 8.11 | Thermograms after 18854.63 fs . . . . .   | 146 |
| A.1  | Running VSV in the compiler debugger mode . . . . .   | 160 |
| A.2  | Example output of the program in Windows during energy calculations. . .  | 162 |
| A.3  | Screen shots of VSV data and Excel converter macro . . . . .  | 180 |

# List of tables

|     |   |     |
|-----|---|-----|
| 3.1 | Dimensioning constants. . . . .   | 39  |
| 3.2 | Sample data for fcc copper . . . . .  | 46  |
| 3.3 | Program output for Table 3.2 input data. . . . .  | 47  |
| 4.1 | SC simulation parameters for fcc copper. . . . .  | 59  |
| 4.2 | Comparison of deterministic and Monte Carlo thermostats . . . . .   | 60  |
| 4.3 | Speed-up comparisons between deterministic and Monte Carlo thermostats  | 60  |
| 6.1 | SC and simulation parameters. . . . .   | 82  |
| 7.1 | The SC simulation parameters used in the simulation test beds for Cu. . . .   | 113 |
| 7.2 | Summary of elastic strain components for a cubic crystal. . . . .   | 122 |
| 8.1 | The file formats used as primary input, intermediate and post-processed<br>outputs in the VSV MD simulation flow. . . . .                             | 135 |
| 8.2 | Summary of simulation results at 0.01K. The comparative literature value<br>are from Kittel [10], Sutton & Chen [14], Behari & Tripathi [23]. . . . . | 142 |
| A.1 | Force calculations on a 3-atom Cu array . . . . .   | 161 |
| A.2 | Potential energy calculation on a 3-atom Cu array . . . . .   | 161 |

# List of C-program codes

|     |   |     |
|-----|---|-----|
| 2.1 | An example of p-thread code spawning 5 threads. . . . .                 | 21  |
| 2.2 | An example of OpenMP parallelization. . . . .                           | 22  |
| 2.3 | Illustration of ordinary and critical OpenMP processes. . . . .         | 23  |
| 2.4 | Syntax for output reduction of parallel processes. . . . .              | 24  |
| 2.5 | An example of MPI parallelization. . . . .                              | 24  |
| 3.1 | Definition of the particle structure used in all simulations. . . . .   | 44  |
| 3.2 | Sample data file for a given VSV simulation. . . . .                    | 47  |
| 3.3 | Typical definition of the constants for a given VSV simulation. . . . . | 48  |
| 4.1 | Actual Windows C code for block timing of parallelized loops. . . . .   | 57  |
| A.1 | Function prototypes used in the header file. . . . .                    | 163 |
| A.2 | Particle management for linked-cell method. . . . .                     | 165 |
| A.3 | Force calculation in linked-cell method. . . . .                        | 165 |
| A.4 | Position and velocity updater in linked-cell method. . . . .            | 166 |
| A.5 | Move particles to their new cells in the linked-cell method. . . . .    | 166 |
| A.6 | Time-based integration function using direct method. . . . .            | 167 |
| A.7 | Time-based integration using modified velocity algorithm. . . . .       | 168 |
| A.8 | Disk file output of final results. . . . .                              | 168 |
| A.9 | Functions for the naive and improved force algorithms. . . . .          | 169 |

---

|   |     |
|---|-----|
| A.10 Functions for the naive and improved force algorithms. . . . .                                   | 170 |
| A.11 Calculation of the $S_k$ intermediate term of the SC potential. . . . .                          | 171 |
| A.12 Force $F_{ij}$ calculation in Sutton-Chen method. . . . .  | 171 |
| A.13 Lennard-Jones 12-6 force $\bar{F}_{ij}$ calculation. . . . .                                     | 172 |
| A.14 Updater functions in the Lennard-Jones 12-6 force $\bar{F}_{ij}$ calculation. . . . .            | 173 |
| A.15 Total ensemble kinetic energy calculation. . . . .   | 174 |
| A.16 Verification of particle contents before simulation. . . . .                                     | 174 |
| A.17 Total ensemble potential energy calculation. . . . .   | 175 |
| A.18 Listing of a typical main program showing output of $U_{tot}$ . . . . .                          | 176 |
| A.19 Code converts VSV output to VMD output for visualization. . . . .                                | 181 |
| A.20 Code calculates the spectral response through FFT of bond length variation<br>with time. . . . . | 184 |

# Chapter 1

## General Introduction

### 1.1 Overview

In the last few decades, computational modelling has become a powerful tool to develop materials by postulation, inference and tuning. In this semi-closed loop approach, material properties that may not be empirically ascertained due to complexity or cost could be evaluated readily [1–6]. A molecular simulation begins with a choice of the best available force field description of the molecular configuration, and culminates in the iterative calculation of the phase space under specified boundary conditions. These boundary conditions expressly convey influences that are external to the particle configuration, such as applied forces, thermostats [7], pressure, and so on.

#### 1.1.1 Computational modelling

The basic computational modelling approach takes the aggregated, macroscopic properties arising as the consequence of their isolated, pair-wise and interactive atomistic dynamics and considering the cumulative effects of these interactions as aggregations by some method that incrementally integrates their equations of motion. Ideally, the force field is expressed

in terms of normalized perturbations (expressed as dimensionless quantities), under the assumption that the internal force field is much stronger compared to the external perturbation. The required phase can then be computed deterministically by MD, or stochastically through Monte Carlo approaches [7, 8]. Ultimately, the results of such computation must be compared with experiment to gauge its accuracy and suitability. Reference is made to a semi-closed loop approach in the context of adjusting either the model or the calculated parameters against known empirical standards, when they exist.

### 1.1.2 Main approaches in computational modelling

There are two broad, general approaches used in simulation [9–18]. The first relies on direct application of the principles of Newtonian mechanics to evaluate overall system trajectory. The second relies on *ab initio* methods, which involve solving quantum mechanical equations from first principles to arrive at overall system behavior. Standard molecular dynamics (SMD) is largely based on the first approach. However, it is important to state that *ab initio* methods may not necessarily exclude aspects of molecular dynamics in their algorithms. For instance, it is possible to use MD in Density Functional Theory (DFT). However, in such applications the calculations become slow. Within the first approach it is customary to derive or define a mathematical force field model which is then simulated using Newtonian mechanics. This mathematical model needs to be as realistic a description of the atomic configuration as possible. The degree of model realism ultimately dictates the degree of success of the simulation. The success of a simulation can be quantified in terms of result accuracy and how efficiently the computational hardware is used. An important and contemporary goal of computation is *parallelization*.

### 1.1.3 Parallelization

Parallelization means the extent to which a single problem could be solved through a scheme of cooperative computation such that results are obtained faster for a given hardware and accuracy specification. To understand this need, one realizes that atomic and molecular systems of macroscopic significance necessarily have extremely large numbers of particles. A macroscopic examination of such a system that is done by evaluating all possible pairwise particle interactions in the system generates vast amounts of data and consumes vast amounts of system time and resources in the process. Some mathematical simplifications, such as specification of a cut-off distance, force field reductions and so on may be done, but such interventions only help to a limited extent. Using a faster processor does not necessarily alleviate the problem immediately, owing to plateauing processor speeds. The issue of computational speed and accuracy is centre stage and continues to highlight the shortcomings of the typical computer for these purposes. In addition, conformity to Moore's law, which predicts a doubling of computational power and speed approximately every two years is no longer assured [19, 20]. In earlier days, conformity to Moore's law relied on improvements in device fabrication techniques which rapidly reduced component interconnect distances on silicon wafer. This distance is expected to reach 5 nm around 2020. However, device computational ability depends on other factors as well, such as precluding quantum mechanical effects arising from the physics of the nanoscale. The high charge mobilities in spaces with high semiconductor component densities have introduced the problem of how efficiently device heat dissipation can be removed [21, 22]. All these factors have contributed to the plateauing processor speeds over the last five years. Clocking frequencies currently stand at approximately 3 GHz.

### 1.1.4 Decomposition and task allocation

Parallelization begins with a decomposition. This means that the problem is divided into smaller ‘chunks’ which are then assigned to a specific processor or core by a task manager. The three most widely used decomposition methods are spatially based, namely particle, force and domain decomposition [23–26]. For practically meaningful simulations these decompositions are still limited by memory allocation, processor coordination and communication between partners. Domain decomposition can present problems for inhomogeneous systems and the cell-task method [27] is suggested as a possible solution to some of the problems. A hybrid decomposition arises when different decomposition methods are used for the same problem. A combination of domain decomposition and thread-based hybrid parallelization based on large vectorization single-instruction multiple data (SIMD) processors and hosting several thousand cores for tens of millions of atoms has recently been reported [21]. A hybrid algorithm ideally achieves parallelization through a task-based approach on smaller sub-domains. Each sub-domain is handled by a team or ‘pool’ of threads running on multiple cores. However, hybrid methods can have issues associated with synchronization in data access in neighbouring cells [28]. A given task manager spawned task may not access the same particle simultaneously. Work in parallelizing MD programs is actively ongoing, for instance [29–33] and others. No one program suits all applications, however. This is because most programs employ rigid parallelization for specific tasks and lack flexibility for newer classes of problems [21]. In spite of this proliferation, smaller code generally accommodates new developments in computing more readily, particularly with respect to scientific computing [21, 34]. Other performance enhancements are concurrent with developments in processor hardware design. Such advancements in general tend to be relegated to dedicated corporations with vast developmental resources. For instance, Peng et al [23] conducted behavioural, instruction-level simulations of supercomputing chip design. This led to an actual, highly portable Godson-T multicore chip of similar attributes [23–25, 35].

Their work addressed the foregoing issues of power efficiency, performance, programming ease, parallelism and platform portability. Such parallelism is said to be *polymorphic* owing to the fact that it is a collective of techniques to achieve the final computational power. For instance, polymorphism can be due to improved chip-local and off-chip communications, fine-grained thread division techniques, better thread synchronization and hardware-based locality awareness. The increased transfer bandwidths, core count and improved cache memory for off-chip transfers realizes considerable performance with the chip on a single motherboard. A supercomputer is easily realized through efficient interconnections of several such motherboards.

### **1.1.5 Statement of the problem**

To achieve dramatic improvements in computational power there had to be a radical shift in the thinking. Today, increases in computational power are due to innovative interconnects of multi-core, multi-processor arrays rather than due to unilateral improvements in processor interconnect distance. Also, more efficient and evolvable algorithms for these hardware arrangements are now driving high performance computation [7, 36, 37]. Demands for realistic computer graphics for gaming is considerable and far outweighs other applications in the public domain. Thus, better force field models and ever more demanding computational algorithms can be hosted on improved platforms and supported by simultaneous developments in Internet and Computer Technology (ICT). The rise of high-performance, super-computing clusters [6] as low-cost supercomputers is often considered to be a secondary revolution in computation. The cost of such super-computers is now falling rapidly alongside the required electrical power requirements [38]. Standalone computation refers to the use of the single computer typified by a single desktop computer. Such a computer, or ‘node’, remains an integral component of any cluster. Standalone does not mean a single-processor or single-core node, but only that the computing node does not feature as part of a distributed

system. A physically isolated, single machine having anywhere from single core to multiple single core processors to multiple processors with multi-cores qualifies this definition. It is important that any talk of overall system efficiency and accuracy must start at the level of the node through careful design. However, a survey of the literature suggests that this is rarely done since most systems have a computational power reliant on brute-force power scaling through added nodes. This introduces a weakness in computation especially during upward hardware-size scaling. Within the parallelization context there needs to be a task manager that efficiently distributes tasks to various nodes and then collects the chunks that represent pieces of the results. The task manager program necessarily handles a vast inter-node and inter-process communications and can easily introduce additional weaknesses in the distributed computational scheme.

### **1.1.6 Objectives of the research**

The primary aim of this research is to create a repeatable and reproducible molecular dynamics simulations framework. Generally, there is a high starting inertia in computation and MD is not exempt. Many reasons are ventured forth for this. What is certain is that unlike experimental research, the current, advanced state of dedicated computing is such that breaking into computation can require a monumental effort. Therefore, this research is aimed ultimately at offsetting these requirements. The standalone node in the foregoing discussion is conceived as a necessary starting point, and developed with performance efficiency and easy scalability in mind to realize a high performance infrastructure. The specific objectives of this research are

- To present a detailed review of the current state of the art in MD simulation techniques, by highlighting the role of the single computer and ways to scale it up towards bigger clusters of such computers.

- To develop and write own code using the C/C++ programming language that concisely implements the mathematical models and algorithms for simulation. The overall program is to be compiled as reusable functions put together in a freely distributable library. The code is to be assisted wherever necessary by applets written in Visual Basic and Matlab.
- The developed software is to focus on a given class of selected fcc metal problems. The calculable problems involve mechanical and energetic parameters. The calculated parameters were decided based on the ready availability of published literature data to aid comparison with the simulation outputs.
- The developed program is to implement parallelization and scalability such that it can be run with minimal modification on multi-core, multi-processor standalone computers across different operating systems.
- To deploy freely the developed program as a free educational and research tool and to apply it to solve actual, current problems relating to metallic systems in a novel and innovative way.
- To apply the developed tools to solve actual physics problems in metallic condensed matter systems for the purpose of improving their understanding, thereby expanding the known body of knowledge.
- To detail the steps to emulate for other metallic systems other than fcc

### 1.1.7 Outline of the thesis

**Chapter 2** provides the rationale for the work contained in this thesis. It begins by reviewing the current approaches in MD simulations. The main aim of the chapter is to emphasize the considerations that must be made to achieve enhanced computational performance in

the general absence of high performance computing resources, as is the case for single, standalone computers in simulation. The chapter is a starting starting point for the interested reader embarking on standalone MD without assuming advanced prior knowledge about the subject. The chapter then gradually guides the reader towards practical, reproducible experiments within a framework, which is developed soon thereafter in this thesis. The framework is required to be extendable to larger computing clusters by a design feature that makes the software scalable with a minimum of reconfiguration onto additionally connected computers. Finally, the chapter outlines the minimum requirements for such a framework to achieve its indicated aims.

**Chapter 3** begins by describing the kinds of particle ensembles that can be investigated using the main outputs of this work. The pairwise-particle potential approach is discussed, as well as how a system of particles is modified through the energy required to embed a subsequent particle into an existing system, resulting in the embedded atom model (EAM) approach. We settle on the Sutton-Chen implementation of the Finnis-Sinclair potentials as being sufficiently descriptive for a class of fcc metallic lattice systems that have a practical relevance. We derive the mathematical models for fcc, with a focus on copper atoms. Finally, we develop the software based on these equations. The subsequent chapters evaluate the performance of the software and then apply it to solve actual dynamical particle problems in physics. The main output of this chapter is the SoftwareX journal article,

**Chapter 4** evaluates the developed software in the standalone setting, and then on a small, four-computer cluster consisting of multi-core, multi-processors. The system size tested consists of 30,261 copper atoms in an arbitrarily defined crystalline lattice. The overall system behaviors are calculated and compared for performance enhancements. In the first case, the energetics of the cluster are determined. In the second case the temperature evolution of the system is followed. In the first step deterministic methods are used. In the second step, a Monte-Carlo approach is used.

**Chapter 5** investigates the performance enhancements of the implemented parallelization of the VSV software on the standalone platform as applied to simulate fcc Cu lattices. The outputs of VSV are compared with the known parameters from the literature. A method to evaluate the performances meaningfully is first developed, and then measurements of execution times of the code blocks were conducted using system timer functions.

The following Chapters 6 and 7 apply the VSV tool and its extensions in Visual Basic and Matlab to enable visualization. Thus, VSV is demonstrated to solve two new, actual and current problems in condensed matter physics. Through the outputs of these subsequent chapters, reasonable explanations of empirically observed phenomena are provided for the first time.

**Chapter 6** presents a standard MD simulation using VSV to investigate phonon propagation on the surfaces and within the bulk of a nanostructured fcc copper lattice. At the present time, the systems reported in the literature and investigated empirically have had large particle numbers, and at temperatures approaching melting point. The dimensions of the investigated systems make them macrostructures. A number of questions are increasingly being asked, such as whether the physics of the nanoscale mirrors that of the macroscale. Also, the effect of the passage of wave-like disturbances on a lattice at much lower than melting temperatures i.e. at lower energies. Such questions are pertinent in the light of the growing interest nanostructures. Such configurations of atoms represent nanoscale metallic connects for nanoscale devices. Through the wave-like propagation of energy in response to an applied temperature differential using energy equipartition that is detectable in VSV, this chapter investigates the relative motions of lattice atoms and suggests an explanation for the effective diffusion using a mechanism other than Brownian motion. In doing so, we provide first evidence of spontaneous structure coalescence. Using VSV simulation results we calculate the diffusion parameters and compare them with parameters from standard diffusion. Finally, this chapter simulates the assimilation and growth of fcc Cu lattices.

**Chapter 7** uses the ability of the developed software to detect dynamical oscillations in the bond lengths as described in the previous chapter. The chapter focuses considerably on detection and following of bond-length oscillations as a mechanism for the dynamical study of the lattice.

**Chapter 8** extends the observed bond-length oscillations into an impulse-oscillation response approach that is eventually used to calculate the elastic constants of the nanostructures. The results are compared with the literature values and the agreements indicate that the approach is reasonable to investigate the phonon dispersion properties of the lattice.

**Chapter 9** presents the concluding remarks pertaining to the research. It also delimits the research but indicates scope and directions of current research questions that could be investigated using the developed tools.

**Chapter 10** lists some of the outputs of the research that have been published in journals and presented at conferences thus far. It also details the manuscripts that have been prepared for publication and are in process.

# References

- [1] Horsch M, Vrabec J, Bernreuther M, Grottel S, Reina G, Wix A, Schaber K, Hasse H. Homogeneous nucleation in supersaturated vapors of methane, ethane, and carbon dioxide predicted by brute force molecular dynamics. *The Journal of Chemical Physics*, 128(164510), 2008.
- [2] Mendeleev MI, Han S, Srolovitz DJ, Ackland GJ, Sun DY, Asta M. Development of new interatomic potentials appropriate for crystalline and liquid iron. *Philosophical Magazine*, 83(35):3977–3994, 2003. doi: 10.1080/14786430310001613264.
- [3] Sutton AP, Chen J. Long-range Finnis-Sinclair potentials. *Philosophical Magazine*, 63(1):139–156, 1990.
- [4] Das A, Ghosh MM. MD simulation-based study on the melting and thermal expansion behaviours of nanoparticles under heat load. *Computational Materials Science*, 101: 88–95, 2015. doi: 10.1016/j.commatsci.2015.01.008.
- [5] van der Walt C, Terblans JJ, Swart HC. Molecular dynamics study of the temperature dependence and surface orientation dependence of the calculated vacancy formation energies of Al, Ni, Cu, Pd, Ag, and Pt. *Computational Materials Science*, 83:70–77, 2014. doi: 10.1016/j.commatsci.2013.10.039.

- 
- [6] Abraham MJ, Murtola T, Schulz R, Pall S, Smith JC, Hess B, Lindahl E. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 101:88–95, 2015. doi: 10.1016/j.softx.2015.06.001.
- [7] Ocaya R, Terblans JJ. Temperature specification in atomistic molecular dynamics and its impact on simulation efficacy. *J. Phys.: Conf. Ser*, 905(012031), 2017. doi: 10.1088/1742-6596/905/1/012031.
- [8] Buchholz M, Bungartz H-J, Vrabec J. Software design for a highly parallel molecular dynamics simulation framework in Chemical Engineering. *Journal of Computational Science*, 2(2):124–129, 2011. doi: 10.1016/j.jocs.2011.01.009.
- [9] Smirnov BM. Energetics of clusters with a face centered-cubic structure. *Zh. Eksp. Teor. Fiz.*, 107:2080–91, 1995.
- [10] Terblans JJ. Calculating the bulk vacancy formation energy ( $E_v$ ) for a Schottky defect in a perfect Cu(111), Cu(100) and a Cu(110) single crystal. *Surf. Interface Anal.*, 33: 767–70, 2002. doi: 10.1002/sia.1451.
- [11] Mattsson TR, Mattsson AE. Calculating the vacancy formation energy in metals: Pt, Pd, and Mo. *Physical Review B*, 66(214110), 2002.
- [12] Sebastian IS, Aldazabal J, Capdevila C, Garcia-Mateo C. Diffusion simulation of CrFe bcc systems at atomic level using a random walk algorithm. *Phys. Stat. Sol. (a)*, 205(6): 1337–1342, 2008. doi: 10.1002/pssa.200778124.
- [13] Jian-Min Z, Fei M, Ke-Wei X. Calculation of the surface energy of fcc metals with modified embedded-atom method. *Applied Surface Science*, 13(7):34–42, 2004. doi: 10.1016/j.apsusc.2003.09.050.

- [14] Griebel M, Knapek S, Zumbusch G et al (Eds.). Numerical simulation in molecular dynamics. *in, Texts in Computational Science and Engineering 5, Springer, Berlin, 5* (ISBN 978-3-540-68094-9), 2007.
- [15] Car R, Parrinello M. Unified approach for molecular dynamics and density functional theory. *Phys. Rev. Lett.*, 55(22):2471–2474., 1985.
- [16] Car R, Parrinello M. The unified approach for molecular dynamics and density functional theory, in simple molecular systems at very high density. *in NATO ASI Series, Series B, Physics, P.P. Loubeyre and N. Boccara (Eds.)*, 186:455–476, 1989.
- [17] Remler DK, Madden PA. Molecular dynamics without effective potentials via the Car-Parrinello approach. *Mol. Phys.*, 70(6):921–66, 1990.
- [18] Tuckerman ME. Ab initio molecular dynamics: basic concepts, current trends and novel applications. *J. Phys.: Condens. Matter*, 2002. URL [stacks.iop.org/JPhysCM/14/R1297](http://stacks.iop.org/JPhysCM/14/R1297).
- [19] Moore GE. Cramming more components onto integrated circuits. *Electronics Magazine*, 1965.
- [20] Moore GE. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1), 1998.
- [21] Mangiardi CM, Meyer R. A hybrid algorithm for parallel molecular dynamics simulations. *arXiv:1611.00075*, 2016. URL [cond-mat.mtrl-sci].
- [22] Mangiardi CM, Meyer R. Molecular-dynamics simulations using spatial decomposition and task-based parallelism. *in: J. Belair et al. (Ed.), Mathematical and Computational Approaches in Advancing Modern Science and Engineering. Springer International, Switzerland*, pages 133–140, 2016.

- [23] Peng L, Tan G, Kalia RK, Nakano A, Vashishta P, Fan D, Sun N. Preliminary investigation of accelerating molecular dynamics simulation on Godson-T many-core processor. *Euro-Par 2010 Parallel Processing Workshops*, 6586:349–356, 2010.
- [24] Peng L, Tan G, Kalia RK, Nakano A, Vashishta P, Fan D, Sun N. Preliminary investigation of accelerating molecular dynamics simulation on Godson-T many-core processor. *Parallel Processing Workshops, Euro-Par 2010: HeteroPar 2010, HPPC 2010, HiBB 2010, CoreGrid 2010, UCHPC 2010, HPCF 2010, PROPER 2010, CCPI 2010, VHPC 2010, Ischia, Italy*, 86021, 2010.
- [25] Peng L, Tan G, Kalia RK, Nakano A, Vashishta P, Fan D, Sun N. Preliminary investigation of accelerating molecular dynamics simulation on Godson-T many-core processor. *Lecture Notes in Computer Science, Euro-Par 2010*, 6586, 2010.
- [26] Bernreuther M, Bungartz H.-J. Molecular simulation of fluid flow on a cluster of workstations. in: *F. Hülsemann, M. Kowarschik, U. Rüde (Eds.), Proceedings of the 18th Symposium Simulationstechnique (ASIM2005), in Frontiers in Simulation, SCS European Publishing House*, pages 117–123, 2005.
- [27] Meyer R. Efficient parallelization of short-range molecular dynamics simulations on many-core systems. *Phys. Rev. E.*, 8(5):053309, 2013.
- [28] Hager G, Wellein G. *Introduction to High Performance Computing for Scientists and Engineers*. Chapman and Hall, 2011.
- [29] Plimpton S. Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.*, 117(1):1–19, 1995.
- [30] Todorov IT, Smith W, Trachenko K, Dove MT. DL POLY 3: new dimensions in molecular dynamics simulations via massive parallelism. *J. Mater. Chem.*, 16(20): 1911–1918, 2006.

- [31] Phillips JC, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel RD, Kalé L, Schulten K. Scalable molecular dynamics with NAMD. . *J. Comput. Chem.*, 26(16):1781–1802, 2005.
- [32] Ackland GJ, D’Mellow K, Daraszewicz SL, Hepburn DJ, Uhrin M, Stratford K. The MOLLY short-range molecular dynamics package. *Comput. Phys. Commun.*, 182(12): 2587–2604, 2011.
- [33] Berendsen HJC, van der Spoel D, van Drunen R. GROMACS: A message-passing parallel molecular dynamics implementation. *Comput. Phys. Commun.*, 91(1):43–56, 1995.
- [34] Needham PJ, Bhuiyan A, Walker RC. Extension of the AMBER molecular dynamics software to Intel’s Many Integrated Core (mic) architecture. *Comput. Phys. Commun.*, 201:95–105, 2016.
- [35] Wang J, Gao X, Li G, Liu Q, Hu W, Chen Y. Godson-3: A scalable multicore RISC processor with x86 emulation. *IEEE Micro.*, 2(29), 2009. doi: 10.1109/MM.2009.30.
- [36] Ocaya R, Terblans JJ. C-language package for standalone embedded atom method molecular dynamics simulations of fcc structures. *SoftwareX*, 5:107–111, 2016. doi: 10.1016/j.softx.2016.05.005.
- [37] R O Ocaya and J J Terblans. Coding considerations for standalone molecular dynamics simulations of atomistic structures. *Journal of Physics: Conference Series*, 905(1): 012018, 2017. doi: 10.1088/1742-6596/905/1/012018. URL <http://stacks.iop.org/1742-6596/905/i=1/a=012018>.
- [38] Dongarra J. Survey of present and future supercomputer architectures and their interconnects. *Paper presented in International Supercomputer Conference, Heidelberg, Germany*, 2004.

# Chapter 2

## Designing a MD simulation testbed

### 2.1 Introduction

For a MD simulation, the architecture of the simulation platform can limit the most that can be achieved in terms of the performance parameters discussed in Chapter 1. A discussion of the common computer architectures on which such simulations are expected to run is therefore necessary, particularly with respect to the processor-memory arrangement.

All low-level operations follow the general sequence: fetch an instruction or data or both, then execute, then store the result back into memory. The sequence is repeated until the end or a branch occurs. A von Neumann type machine arises when the fetches occur along the same path - first the instructions, then the data. In the Harvard type the fetches occur along different paths prior to execution. The latter type of machine therefore has a pre-fetch ability and can execute instructions concurrently with fetches. This instruction/data *pipelining* substantially enhances performance and is implemented on most processors today. Pipelining has a number of technical challenges that arise mainly during conditional branching. However, these are not in the present scope.

This chapter has three main goals. First, it reviews the state of the art of MD simulations on various hardware and software platforms. Second, it outlines the achievable targets for

our own contributions in MD on standalone machines. Thirdly and overall, it describes a testable, parallelized testbed program that addresses the performance issues raised in Chapter 1 and enables the actual simulations in the research.

### **2.1.1 Classification of parallelization paradigms**

The typical MD problem falls into two broad categories:

1. ‘embarrassingly’ parallel - which implies that it is readily separated into identifiable, unique tasks that may then be executed separately. In other words, the execution of one process does not rely on the execution of another.
2. ‘serial’ - which implies that it cannot be split into independent sub-problems and the results of one computation drive the next. Such a problem requires either inter-process communication, or completion of one process first.

However, a clear demarcation into either category may not always be possible for the problem. The solution could then include elements of each category. However, most parallelization occurs at the level of the iterations (loop summations) where the force fields are applied. The other parts of the typical program are the user and output interfaces. These parts of the code are concerned with loading particle structures into memory, and can implement iterations that allow the particles in the configuration to relax to an equilibrium decided by the (new) boundary condition (called ‘set up’ time), such as applied temperature. These parts of the code can take inordinately large multiples of simulation time-step, but typically occur only once per simulation and are therefore not considered in performance benchmarking [1].

### **2.1.2 Coding versus proprietary software**

The goal of writing simulation software is to assure functionality, accuracy and reproducibility of empirical results, while being inherently parallelizable. For all the material science

problems that could be solved by MD techniques, a code may already exist in some form or other. However, the heuristic or ‘black box’ nature of such software obscures its inner workings. For other software the code may not be general enough for all problems thereby making it inflexible towards new problems. Also, advancements in computer architecture and computational methods are often taken advantage of by highly modular, more transparent codes. Placing ‘open’ code into the public domain under various licenses, such as GNU and others can lead to a collaborative effort that improves the code by virtue of the peer-review process. Therefore, one needs to carefully consider the nature of the problem to be solved. For instance, it could be asked what the expected solution longevity is, the availability of similar software, the complexity of the problem particularly in the absence of similar software, the technical resources available in terms of development time, support, and so on. To illustrate the dilemma, for density functional theory (DFT) calculations [2] many free software exist but with varied strengths and weaknesses with none being suited to all tasks. It has been suggested that for an academic undertaking writing own software can produce a software that is small, transparent, fast, modular and potentially extensible to new problems while being highly parallelizable.

## 2.2 General tools

Considering possible parallelization, a successful standalone solution is one that is highly scalable in size for the same or improved accuracy. Scalability is defined as the ability to increment the system in terms of size and computational power by physically adding similar systems to an existing system while adjusting only a small set of reconfiguration parameters. A ‘hybrid’ system is possible and often desirable by making use of subsystems comprising of heterogeneous hardware and operating software. Such interconnections define a ‘farm’, with interconnection compliance ensured through communication protocol standards. In the ideal

case the specific hardware requirements are less important. Lower overall system cost and ease of reconfiguration are usually benefits of systems that have homogeneous components.

### 2.2.1 Parallelizable tools

For parallelization, the fundamental questions are:

1. How can the full computational power of a standalone computer having  $P$  processors and  $C$  cores be exploited?
2. Is this set  $\{P, C\}$  available by default by virtue of the unit(s) being on?
3. How does one benchmark the performance of this set?

At present, these questions remain largely unanswered. Several studies have attempted to correlate the performance of the system as a function of the size in terms of the set  $\{P, C\}$ . For instance, Peng et al [3] measure the speedup on  $p$  cores for a problem of fixed size as  $S(p)=T_1/T_p$ , which is the ratio of the execution time on one core compared to the execution time on the  $p$  cores. The strong-scaling parallel efficiency  $E_p$  is then  $E(p)=S(p)/p$ . They also monitor the performance metrics within cores for their quad-core simulations using Intel's VTune Performance Analyzer. Others, like Pal et al. [4] discuss the bottlenecks in parallelized MD computations and a hybrid algorithm using MPI with OpenMP threads for problems associated with the embedded atom model and Morse potentials. They contrast their study with similar studies using LAMMPS and find that their method produces enhanced performance. In attempting to provide clarity on these questions to a degree of practicality, we begin by outlining some possible tools for the task.

Many parallelizable tools are inherent or implicit in the programming environment and allow migration to parallel computation with relative ease. There are also tools that allow benchmarking of system performance relative to a standard. For instance, LINPACK [5, 6] compares the time taken to co-factorization of a real, dense matrix against a standard. Other

computational tasks that have been used as tests are solvers of ordinary differential equations (ODEs), Fast Fourier transform (FFT), sparse matrix algorithms and others. Therefore, to be meaningful, the true computational power of a system is a holistic aggregate of several benchmarks. In this research all the codes are written in the C-programming language. Other third-party packages such as MATLAB are gradually moving towards parallel computing by adopting a different processor-memory model [7], especially for larger programs. Most programs achieve parallelism in one of three ways. These are:

- multithreaded parallelism (MP), where several simultaneous instruction streams are generated and executed on the above set  $\{P, C\}$ ,
- distributed computing (DC), where several instances of the same program run independently on separate computers with separate memories, and
- Explicit Parallelism (EP), where the problem is solved using parallelized loops, distributed arrays that are decomposed for the set  $\{P, C\}$ .

### 2.2.2 Threads and message passing

The C-language is a standardized, highly portable development language. This accounts for its popularity as the preferred parallelization language. However, newer contenders modeled on its strengths are becoming commonplace. Many of these, such as CUDA, target specific processors such as GPUs. The success of C is due to a number of factors. First, it allows easier definition and manipulation of complex data structures that closely resemble collections of structured ensemble particles. The basic data structure typically consists of a particle type object with directly accessible attributes such as mass, position and velocity. Entire collections of such particles, which represent particle configurations within a domain, can be manipulated just as readily as its individual particles through dedicated functions. Second, the use of pointers within the language introduces flexibility and modularity. In

this way individual particles or entire collections of particles can be exchanged between different sections of the code with ease. An instance where this flexibility is called for is during particle migration, where a particle is added or deleted from a collection. Third, the language is fast and has a number of excellent heap memory management functions, which coupled with the concept of ‘threads’, allows implementation of multi-threaded parallelism. Here, a single problem ‘spawns’ multiple, dynamically managed concurrent but independent sub-programs, i.e. threads. Since threads can access shared memory, careful synchronization and management of threads is needed to limit unwanted thread interaction. This averts ‘collisions’ and ‘racing’ conditions that mar the output of the simulation.

Threads are well-established in the C-programming language and are implemented in several standards. The POSIX or p-thread standard is defined in the “pthread.h” library. The code in Listing 2.1 spawns 5 threads using this approach. Parallelization can also be implemented through distributable multiprocessing application programming interfaces (MAPIs), such as Open Multiprocessing API (OpenMP) and Message Passing Interfaces (MPIs). Some common, freely available MPIs that are a cross between p-threads and OpenMP are OpenMPI (not to be confused with OpenMP), MPI, MPICH2, LAM-MPI.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #define NTHREADS 5
5 void *myFun(void *x){
6     int tid;
7     tid = *((int *) x);
8     printf("Executed thread: %d!\n", tid);
9     return NULL;
10 }
11 int main(int argc, char *argv[]){
12     pthread_t threads[NTHREADS];
```

```
13  int thread_args[NTHREADS];
14  int rc, i;
15  for (i=0; i<NTHREADS; ++i)
16  {
17      thread_args[i] = i;
18      printf("spawning thread %d\n", i);
19      rc = pthread_create(&threads[i], NULL, myFun, (void *)&thread_args
20      [i]);
21  }
22  /* wait for threads to finish */
23  for (i=0; i<NTHREADS; ++i) {
24      rc = pthread_join(threads[i], NULL);
25  }
26  return 1;
}
```

Listing 2.1 An example of p-thread code spawning 5 threads.

### 2.2.3 Open multiprocessing programming

A significant amount of time in complex, repetitive calculations is spent executing loop statements. These iterations are readily parallelized using OpenMP by including the `#pragma` compiler directive followed by the code to be parallelized, as shown in Listing 2.2.

```
1  #include <stdio.h>
2  #include <omp.h>
3  int main(int argc, char **argv){
4      int i, thread_id, nloops;
5
6      #pragma omp parallel private(thread_id, nloops){
7          nloops = 0;
8      #pragma omp for
9          for (i=0; i<1000; ++i) ++nloops;
```

```
10     thread_id = omp_get_thread_num();
11     printf("Thread %d, loop iteration %d.\n", thread_id, nloops );
12 }
13 return 0;
14 }
```

Listing 2.2 An example of OpenMP parallelization.

The same directive can be used to specify the aspects of critical code i.e. code that is common to all the threads, as shown in Listing 2.3.

```
1 #include <stdio.h>
2 #include <omp.h>
3 int main(int argc, char *argv []){
4     int i, thread_id;
5     int glob_nloops, priv_nloops;
6     int glob_nloops = 0;
7
8     #pragma omp parallel private(priv_nloops, thread_id); {
9         priv_nloops = 0;
10        thread_id = omp_get_thread_num();
11        #pragma omp for
12        for (i=0; i <100000; ++i) ++priv_nloops;
13        #pragma omp critical{
14            printf("Thread %d is adding its iterations (%d) to sum (%d)",
15                thread_id, priv_nloops, glob_nloops);
16        }
17        printf("total nloops is now %d.\n", glob_nloops);
18        return (0);
19 }
```

Listing 2.3 Illustration of ordinary and critical OpenMP processes.

The `critical` keyword manages shared aspects between parallelized sections. This is implicitly serial but has the main advantage that it eliminates collision problems e.g. when code sections attempt simultaneous access of a shared variable. The process of combining the outputs of several smaller parallel sections is known as *reduction*, and is achieved in OpenMP using the `reduction` keyword. The syntax is shown in Listing 2.4. There are other keywords that are associated with the `#pragma omp` directive, but these can be found in the OpenMP documentation [8].

```
1 #pragma omp parallel private(priv_nloops , thread_id) reduction(+:  
   glob_nloops)
```

Listing 2.4 Syntax for output reduction of parallel processes.

Listing 2.5 shows the main aspects of parallelized programs used in the present work in the calculation of cluster energetics. It is well-suited to run on a Linux cluster.

```
1 // mpicc go_mpi.c -o go_mpi  
2 // mpirun -n 4 go_mpi  
3 #include <stdio.h>  
4 #include <mpi.h>  
5 int main(int argc , char *argv []) {  
6     int myrank , nprocs ;  
7  
8     MPI_Init(&argc , &argv);  
9     MPI_Comm_size(MPI_COMM_WORLD, &nprocs);  
10    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);  
11    printf("This is node %d of %d\n", myrank , nprocs);  
12    MPI_Finalize();  
13    return 0;  
14 }
```

Listing 2.5 An example of MPI parallelization.

### 2.2.4 MPI programming

The Message Passing Interface (MPI) standard has several implementations that support cross-platform multiprocessing. With MPI, the program assigns a collection of computational resources such as memory, and easily synchronizes all processes on identically handled nodes. The interface does not distinguish between the elements of  $\{P, C\}$ , but simply enumerates each additional element as a computational resource to achieve scaling. Some supercomputers today rely on such an arrangement, which is attractive because it operates well across clusters.

### 2.2.5 GPU computing

Graphical Processing Units or GPUs, because of their fast Reduced Instruction Set Computing (RISC) architectures and lower power consumption, are increasingly being used to implement compute nodes [9–11]. GPUs are specialized at high-speed handling of certain operations that involve massive amounts of data. Many GPUs have multiple cores and are well suited to the calculations that are commonplace in physics and engineering. Today, collections of GPUs on “blade motherboards” can be clustered readily to implement low-cost supercomputers. This is the preferred route for universities. For any scaled system the overall electrical power supply and ventilation needs must be considered carefully.

Brown et al. [12, 13] discuss some issues of porting a large MD program intended purely for CPUs to parallel hybrid machines based on GPUs. By using an accelerated, CPU/GPU balanced load programming model within an existing LAMMPS MD program, they demonstrate hybrid decomposition on CPU/GPU combinations, under short-range force simplifications. Balancing program load between CPUs and GPUs may be necessary because unlike CPUs, many GPUs cannot yet achieve peak floating-point performance in terms of wall-clock time. This disadvantage arises from the GPU’s high optimization of certain specific operations in its instruction set in comparison to other instructions. Also, having some operations run on CPUs can minimize the amount of coding required for functions

executed on GPUs in a hybrid system [12–16]. For instance, Brown et al. [12] show that double precision performance can be poorer than single or mixed precision performance in such clusters, indicating that further optimization is required. Others have argued that increased modularity is advantageous for a scientific software that can be evolved [17].

### 2.2.6 Cloud virtualization

Clouds arose from the need to provide safe data storage for large corporations. The capabilities of clouds have grown to include number crunching and computation on ‘big data’, such as are found in financial markets, weather forecasting, fraud detection, Internet trade, drug research, and other scientific computing. Consequently, they are now referred to as computing clouds. Alongside traditional storage, the compute feature may be supplied free of charge but functionally limited, for educational purposes. At a cost, the full potential of the cloud could be available to a client. The cloud approach has the main advantage of minimal user investment of hardware and technical know-how. However, security and data privacy remain a primary concern for organizations over intellectual property of the research.

An important feature of the cloud is ‘virtualization’, which is the ability to define remote, virtual instances of interconnected, high-performance compute environments from the a graphical interface on the user’s machine. At present, several organizations permit virtualization, such as Yellow Circle [18], Microsoft Cloud (Azure) [19], Google Cloud [20]. Many have provisions for free virtualization for educational purposes. Cloud virtualization is still in its infancy and faces a number of limitations, such as inherent compute delays introduced by virtual machine monitors. Some solutions have been suggested, for instance by Ren et al [21], who propose a light-weight supervisory infrastructure called nOSV. It is likely that in the next few years cloud computing will be a *de facto* starting point for most scientific computing.

### 2.2.7 Specifications of the standalone testbed

The initial standalone machine in this research is a Dell Optiplex 3010 computer. The hardware specifications are:

- 3<sup>rd</sup> generation i5-3470 3.2 GHz processor with 6 MB of L3 cache, *no hyperhtreading*
- 4 cores
- 8 GB of 1.6 GHz DDR3 SDRAM
- AMD Radeon HD 7470 with 1 GB DDR3 SDRAM integrated GPU
- 16-bit PCI Express bus CPU-GPU interconnect
- built-in *K*-technology not activated. If active, permits single core overclocking with other three off [22]. Therefore, in this work it offers no benefits.
- 500 GB hard drive storage to hold initial particle data and boundary conditions and final storage of simulation output. In the interest of speed, during a simulation no intermediate hard drive access is done. All intermediate storage is done in SDRAM.

The software specifications are:

- Windows 10 and Ubuntu Linux 14.0 dual-boot
- Microsoft Visual C++ 2008 Express C-language compiler within Windows
- gcc C-language compiler within Ubuntu Linux
- Intel VTune Performance analyzer (Windows)

Figure 2.1 shows the layout of the scalable cluster. In the figure, there are four compute nodes comprised of standalone machines. Once the cluster is set up, each computer is enumerated as part of the compute set {P, C}.

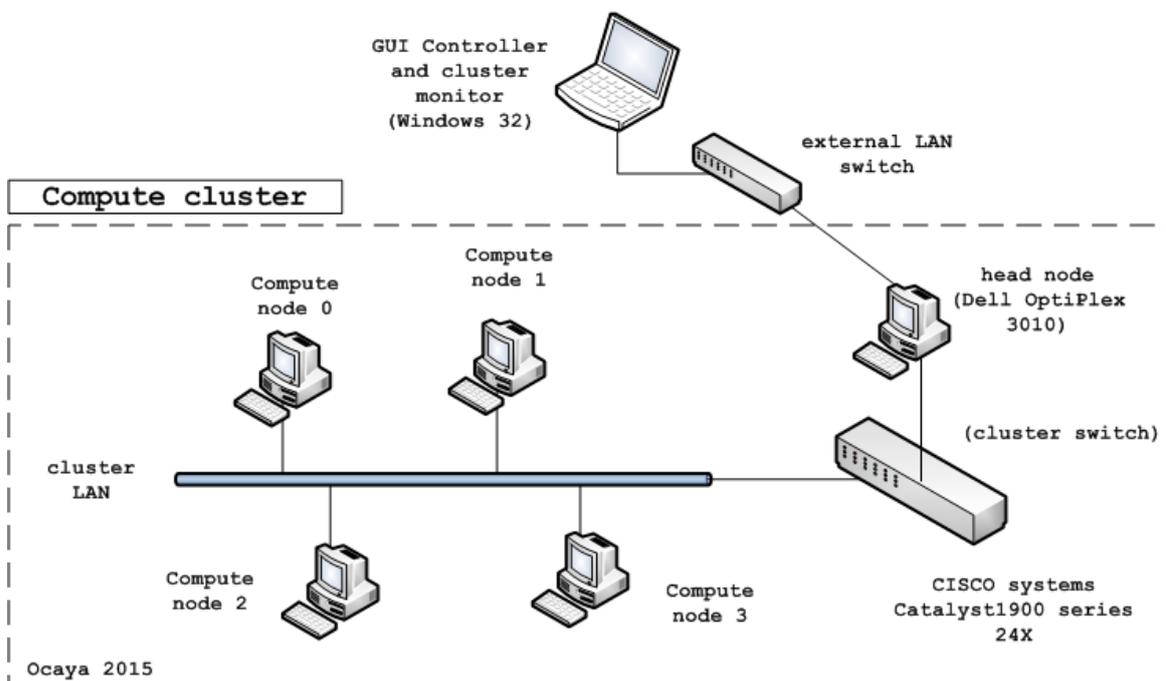


Fig. 2.1 Layout of the scalable compute cluster with heterogeneous standalone machines.

## 2.3 Chapter summary

This chapter has outlined the platform considerations for a computational research in molecular dynamics. It begins by surveying the current state of the art, such as what it takes to achieve parallelism and efficiency, and weighs the pros and cons of writing own code as opposed to using third-party programs for MD. The ultimate goal of the chapter is to outline a real, scalable computing arrangement that exploits the compute abilities of single machines or heterogeneous clusters of single machines. In short, Chapter 2 describes the platform for computational ‘experiments’ of the rest of this thesis with respect to computational performance as discussed above. Chapter 3 presents the mathematical basis for the class of problems investigated computationally in this research.

# References

- [1] Mangiardi CM, Meyer R. A hybrid algorithm for parallel molecular dynamics simulations. *arXiv:1611.00075*, 2016. URL [cond-mat.mtrl-sci].
- [2] Jain A, Shin Y, Persson KA. Computational predictions of energy materials using density functional theory. *Nature Reviews Materials*, 1(15004), 2016. doi: 10.1038/natrevmats.2015.4.
- [3] Peng L, Tan G, Kalia RK, Nakano A, Vashishta P, Fan D, Sun N. Preliminary investigation of accelerating molecular dynamics simulation on Godson-T many-core processor. *Euro-Par 2010 Parallel Processing Workshops*, 6586:349–356, 2010.
- [4] Pal A, Agarwala A, Raha S, Bhattacharya B. Performance metrics in a hybrid MPI–OpenMP based molecular dynamics simulation with short-range interactions. *J. Parallel Distrib. Comput.*, 74:2203–2214, 2014.
- [5] Petitet A, Whaley RC, Dongarra J, Cleary A. HPL - a portable implementation of the high-performance Linpack benchmark for distributed-memory computers. *Document on Internet*, 2016. URL <http://www.netlib.org/benchmark/hpl/>.
- [6] Top500. The Linpack Benchmark. *Document on Internet*, 2016. URL <https://www.top500.org/project/linpack/>.
- [7] Cleve M. Parallel MATLAB: Multiple processors and multiple cores. *The MathWorks News & Notes*, 2007.

- 
- [8] OpenMP Review Board. OpenMP Application Programming Interface. *OpenMP*, 2015. URL <http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf>.
- [9] Govender N, Wilke DN, Kok S, Els R. Development of a convex polyhedral discrete element simulation framework for NVidia Kepler based GPUs. *Journal of Computational and Applied Mathematics*, 270:386–400, 2014. doi: 10.1016/j.cam.2013.12.032.
- [10] Govender N, Wilke DN, Kok S, Els R. Collision detection of convex polyhedra on the Nvidia GPU architecture for the discrete element method. *Applied Mathematics and Computation*, 267:810–829, 2015. doi: 10.1016/j.amc.2014.10.013.
- [11] Govender N, Rajamani RK, Kok S, Wilke DN. Discrete element simulation of mill charge in 3D using the Blaze-DEM GPU framework. *Minerals Engineering*, 79: 152–168, 2015. doi: 10.1016/j.mineng.2015.05.010.
- [12] Brown WM, Wang P, Plimpton SJ, Tharrington AN. Implementing molecular dynamics on hybrid high performance computers - short range forces. *Computer Physics Communications*, 182(4):898–911, 2011. doi: 10.1016/j.cpc.2010.12.021.
- [13] Brown WM, Kohlmeyer A, Plimpton SJ, Tharrington AN. Implementing molecular dynamics on hybrid high performance computers - particle-particle particle-mesh. *Computer Physics Communications*, 183(3):449–459, 2012. doi: 10.1016/j.cpc.2011.10.012.
- [14] Stone JE, Phillips JC, Freddolino PL, Hardy DJ, Trabuco LG, Schulten K. Accelerating molecular modeling applications with graphics processors. *J. Comput. Chem*, 28: 2618–2640, 2007.
- [15] Schmid N, Bötschi M, Van Gunsteren WF. A GPU solvent-solvent interaction calculation accelerator for biomolecular simulations using the gromos software. *J. Comput. Chem.*, 31:1636–1643, 2010.

- 
- [16] Hampton S, Alam SR, Crozier PS, Agarwal PK. Optimal utilization of heterogeneous resources for biomolecular simulations. *in: Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2010*, 5644896, 2010.
- [17] Buchholz M, Bungartz H-J, Vrabec J. Software design for a highly parallel molecular dynamics simulation framework in Chemical Engineering. *Journal of Computational Science*, 2(2):124–129, 2011. doi: 10.1016/j.jocs.2011.01.009.
- [18] Yellow Circle. Yellow Circle Virtualization Cloud. *Document on Internet*, 2016. URL <https://mylab.yellowcircle.net>.
- [19] Microsoft Corporation. Microsoft Azure Virtualization Cloud. *Document on Internet*, 2016. URL <https://azure.microsoft.com/en-us/>.
- [20] Google. Google Cloud. *Document on Internet*, 2016. URL <https://cloud.google.com/>.
- [21] Ren J, Qi Y, Dai Y, Xuan Y, Shi Y. nosv: A lightweight nested-virtualization VMM for hosting high performance computing on cloud. *The Journal of Systems and Software*, 124:137–152, 2017. doi: 10.1016/j.jss.2016.11.001.
- [22] Intel Corporation. Hyperthreading and k-technology. *Document on the Internet*, 2016. URL <http://ark.intel.com/products/65703>.

# Chapter 3

## Development of the simulation software

### 3.1 Introduction

This chapter describes the development of the Velocity Störmer-Verlet software, henceforth abbreviated VSV, targeting the testbed hardware described in Chapter 2. It was written to carry out fast molecular dynamics (MD) calculations on the mechanical evolutions of large ensembles of well-defined particles with masses, positions and velocities in a known potential energy field. In mechanics, particle ensembles may be defined uniquely using parameters that convey size, energy, and so on. Such parameters may be volume, pressure, force, area, particle number, temperature, etc.

#### 3.1.1 The canonical ensemble

Specifically, statistical mechanics defines the *canonical* ensemble as a statistical arrangement that represents possible states of a mechanical system that is in thermal equilibrium with a heat bath at some fixed temperature,  $T_b$ . The equilibrium is not necessarily static, but can be dynamic, implying steady energy exchange between the heat bath and the system. Figure 3.1 illustrates the canonical ensemble, defined as a system having  $N$  particles, volume  $V$  and

temperature  $T$ , and is abbreviated NVT. The bath, although drawn bounded, is taken as an infinite energy source.

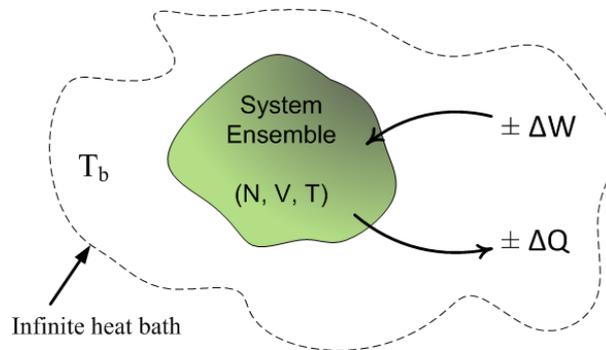


Fig. 3.1 The canonical ensemble.

## 3.2 Defining a particle system

In dynamical simulations of mechanical systems, the main, essential ingredient is a model for the physical system. Choosing a model amounts to *defining* a potential energy function that best describes the system. Therefore, it is not enough to merely define an ensemble, which in reality only specifies the boundary conditions under which the model operates. The model specifies the physical laws that govern the evolution of states between boundary conditions.

The contemporary approach in MD, as with the developed VSV code, the potential energy function  $V$  is defined in terms of the position vector set  $\{\vec{r}_i\}$  for all  $N$  particles, i.e.  $i=\{1, 2, \dots, N\}$ . Thus

$$V = V(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_N). \quad (3.1)$$

The potential must be invariant to translation and rotation.

### 3.2.1 Force in atomic particle systems

Force ( $\vec{F}$ ) is derived as the gradient of potential energy with respect to the particle displacements, i.e.

$$\vec{F} = -\nabla V(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_N). \quad (3.2)$$

This form of force specification implies energy conservation i.e. the existence of the Hamiltonian, E:

$$E = K + T, \quad (3.3)$$

where total kinetic and potential energies are respectively K and T. The simplest choice of  $V$  relies on pairwise interactions between any two system particles  $i$  and  $j$  and their relative positions, i.e.

$$V(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_N) = \sum_i^N \sum_{j>1}^{N-1} \phi(|\vec{r}_i - \vec{r}_j|). \quad (3.4)$$

While intuitive and relatively easy to implement, there are two fundamental problems with this two-body modeling approach, namely:

1. It is computationally demanding as system size increases.
2. It does not properly describe important particle systems such as metals, semiconductors, and new materials.

## 3.3 Embedded Atom Modeling

Two-body potentials that are still either in use or have historical significance are the Lennard-Jones (LJ), Stillinger-Weber, Buckingham, Long-range Coulomb, Tersoff, Brenner, Sutton-Chen, Morse potentials, and many more. These potentials mostly describe particle interactions through the interplay of attractive and repulsive particle-particle forces. For instance, when the particles get to within a specified distance of each other repulsion dominates;

beyond a certain separation, attraction dominates. No one potential models all systems equally well and the best one for a particular system is arguable. The LJ potential, commonly referred to as the 12-6 potential is, for instance, known to correctly model atomic systems having closed-shell atom interactions (e.g. rare gases such as argon). However, it fails for closed-shell systems where the localized bonds are strong (e.g. for covalent bonds), or even where there is a strong electron delocalization “sea” (e.g. metals). Therefore in metals, the LJ potential describes surface relaxation poorly.

### 3.3.1 The Finnis-Sinclair approach

To address the limitations of the two-body approach, the *embedded atom model* (EAM) method was developed. The Finnis-Sinclair (FS) potential [1] is an EAM method potential that is a composite of a two-body potential to model repulsion, but with the advancement of a *density of state* function  $\rho_i$  to model attraction. The particles are presumed to be in the fcc crystalline arrangement. This additional term represents the energy needed to embed an atom  $i$  (hence the term “embedded”) into the array in terms of the receiving host electron density. The total system potential energy,  $U_{tot}$  (or T in the Hamiltonian) is then

$$U_{tot} = \frac{1}{2} \varepsilon \sum_i^N u_i, \quad (3.5)$$

where for  $n > m$ ,

$$u_i = \frac{1}{2} \varepsilon \sum_{j \neq i}^N \left[ V(r_{ij}) - c \rho_i^{1/2} \right], \quad V(r_{ij}) = \left( \frac{a}{r_{ij}} \right)^n, \quad \rho_i = \sum_{j \neq i}^N \left( \frac{a}{r_{ij}} \right)^m. \quad (3.6)$$

The constant  $\varepsilon$  is a dimensionless energy scaling parameter,  $a$  is the lattice constant and  $c$  is a fitting parameter.

### 3.3.2 The Sutton-Chen form of the Finnis-Sinclair potential

The VSV code implements the Sutton-Chen (SC) form [2] of the Finnis and Sinclair (FS) potential [1]. FS described the force  $\bar{F}_i$  on a system particle  $i$  in terms of the net effect of pairwise repulsive-cohesive forces with all the other particles in the system and the local charge density  $\rho_i$  around the particle. The latter conveys the attractive effects of a perturbing or ‘embedded’ atom. The total FS potential energy has the form:

$$U = \sum_i^N u_i, \quad (3.7)$$

where

$$u_i = \frac{1}{2} \varepsilon \sum_{j \neq i}^N \left[ V(r_{ij}) - c \rho_i^{1/2} \right], \quad \rho_i = \sum_{j \neq i}^N \left( \frac{a}{r_{ij}} \right)^m \quad \text{for } n > m, \quad (3.8)$$

where  $a$  is the lattice constant,  $\varepsilon$  and  $c$  are energy scaling and fitting parameters respectively. The pairwise interaction potential between particles  $i$  and  $j$  that are separated a distance  $\bar{r}_{ij}$  is given by

$$V(r_{ij}) = \left( \frac{a}{r_{ij}} \right)^n. \quad (3.9)$$

Sutton-Chen (SC) [2] introduced an embedding energy  $E(\bar{\rho}_i)$  term into the FS potential to rather describe the energetic contribution of the local charge density, hence

$$U = \frac{1}{2} \sum_{ij}^N V(r_{ij}) + \sum_i^N E(\bar{\rho}_i). \quad (3.10)$$

Equation (3.10) can be written

$$U = \varepsilon \sum_{i=1}^N \left( \sum_{j=i+1}^N \left( \frac{\sigma}{r_{ij}} \right)^n - c \sqrt{S_i} \right), \quad (3.11)$$

where

$$S_i = \sum_{j=1, j \neq i}^N \left( \frac{\sigma}{r_{ij}} \right)^m. \quad (3.12)$$

The force,  $F = -\bar{\nabla}V(r)$ , on particle  $i$  is then

$$\bar{F}_i = \varepsilon \sum_{j=1, j \neq i}^N \left[ n \left( \frac{\sigma}{r_{ij}} \right)^n - \frac{cm}{2} \left( \frac{1}{\sqrt{S_i}} + \frac{1}{\sqrt{S_j}} \right) \left( \frac{\sigma}{r_{ij}} \right)^m \right] \frac{\bar{r}_{ij}}{r_{ij}^2}. \quad (3.13)$$

where  $\bar{r}_{ij}$  is the vector between particles  $i$  and  $j$ , i.e.

$$\bar{r}_{ij} = \bar{x}_j - \bar{x}_i. \quad (3.14)$$

VSV code employs standard MD to evolve the position and momentum of each particle at each time step [3].

### 3.3.3 MD simulation using adatoms

The EAM approach, because it fundamentally involves particle embedding, presents the unique opportunity to add or subtract atoms arbitrarily to existing arrangement of atoms and therefore to investigate additive/subtractive perturbations on the global system. The added or subtracted atom or atoms are called ‘adatoms’. In this way, the effects of missing atoms, substitution by different atoms, arbitrary configurations may be examined in simulations, something that is impossible to do experimentally. In essence, this is equivalent to the postulation of defects in the crystalline array. The study of such defects can lead to enhanced understanding of many macro properties of the material i.e. its thermal properties by phonon coupling, melting point, mechanical strength, and so on. These results may then be interpreted in the context of established theories.

Section 3.4 shows how the VSV program can be used to calculate the global minimum in transition metal clusters using the foregoing Sutton-Chen potential in the EAM method.

### 3.4 Velocity Störmer-Verlet integration

A version of this time evolution technique [4] is implemented in the program. It is commonly used to discretize the Newtonian equations of motion [4], and has  $O(\delta t^2)$  discretization error. In the discretized notation  $\bar{p}_i^n := \bar{p}_i(t_n)$  which describes the vector  $\bar{p}$  sampled at the discrete  $n$ -th time interval, where  $\bar{p} = \{\bar{x}, \bar{v}, \bar{F}\}$ , it follows that the approximation of  $\bar{p}_i^{n+1}$ , i.e. on the next or  $(n+1)$ -th time interval or  $(t + \delta t)$  gives the new positions

$$\bar{x}_i^{n+1} = \bar{x}_i^n + \delta t \bar{v}_i^{n+1} + \frac{\bar{F}_i^n}{2m_i} \delta t^2. \quad (3.15)$$

Similarly, the next velocities are

$$\bar{v}_i^{n+1} = \bar{v}_i^n + \left( \frac{\bar{F}_i^n + \bar{F}_i^{n+1}}{2m_i} \right) \delta t. \quad (3.16)$$

#### 3.4.1 Determination of phase space

The overall goal of MD simulation is to aggregate the properties of microscopic particle collections into singular parameters that can be used to explain the macroscopic properties of the material. The phase space is an important and completely macroscopic description of a physical system using all its possible states; conversely, each possible state is represented by a unique point in the description. In classical dynamical systems, the canonical phase space representation considers the momentum and position evolution. Therefore, the preceding chapters fall into this category and the VSV program is equally suited to such systems as well. In order to reach such an aggregate, the simulation ideally starts by expressing parameters as dimensionless quantities.

### 3.5 Dimensionless equations

Dimensionless equations in simulations are used for three main reasons. To:

1. prevent very small or very large numbers associated with input or output quantities from producing unwieldy calculations in a simulation. This offsets the storage needs for such numbers.
2. provide a means of quickly determining the relevant quantities and coefficients for system evolution.
3. have a simulation that is invariant of the measurement system but only descriptive of the physical system.

Reducing a variable to a dimensionless equivalent requires scaling it down using a known value reference variable of equal dimension. In other words,

$$\text{new variable (dimensionless)} = \frac{\text{old variable (dimensioned)}}{\text{constant reference (same dimensions)}}$$

The new, dimensionless variable ( $n'$ ) may be written symbolically as  $n' = n/\tilde{n}$ , where  $n$  is the old variable and  $\tilde{n}$  is the reference. Choosing the characteristic quantities according to Table 3.1 allows all other reference quantities such as time, force, pressure, temperature to be written as reduced variables.

Table 3.1 Dimensioning constants.

| Variable | Quantity      | Reference             |
|----------|---------------|-----------------------|
| length   | $\sigma$      | $\tilde{\sigma}$      |
| mass     | $m$           | $\tilde{m}$           |
| energy   | $\varepsilon$ | $\tilde{\varepsilon}$ |

### 3.5.1 Time

Here, we use the example of time to illustrate the reduction to dimensionless variable. Since  $\varepsilon \equiv m \times v^2$ , and  $v \equiv \sigma/t$  i.e. energy is equivalent to the product of mass and square of velocity, while velocity is equivalent to displacement divided by time, it follows that

$$\varepsilon \equiv m \times \left(\frac{\sigma}{t}\right)^2. \quad (3.17)$$

Rearranging Equation (3.17) leads to

$$t \equiv \sqrt{\frac{\sigma^2 m}{\varepsilon}}. \quad (3.18)$$

Then, since  $\sigma = \tilde{\sigma}\sigma'$ ,  $m = \tilde{m}m'$  and  $\varepsilon = \tilde{\varepsilon}\varepsilon'$ , direct substitution into Equation (3.18) leads to the dimensionless time scale:

$$t' = \frac{t}{\tilde{\alpha}}, \quad \text{where} \quad \tilde{\alpha} = \sqrt{\frac{\tilde{\sigma}^2 \tilde{m}}{\tilde{\varepsilon}}}, \quad (3.19)$$

where the time scaling is denoted  $\tilde{\alpha}$  rather than  $\tilde{t}$ . Similar arguments can be followed for the remaining variables. The next sections list the reduced variables that are useful for the phase space calculation in the VSV toolkit.

### 3.5.2 Velocity and acceleration

Since  $v \equiv \sigma/t \equiv (\tilde{\sigma}\sigma')/(\tilde{\alpha}t')$ , it follows that

$$\frac{\partial \bar{x}}{\partial t} = \frac{\tilde{\sigma}}{\tilde{\alpha}} \frac{\partial \bar{x}'}{\partial t'}, \quad (3.20)$$

and acceleration,

$$\frac{\partial^2 \bar{x}}{\partial t^2} = \frac{\tilde{\sigma}}{\tilde{\alpha}^2} \frac{\partial^2 \bar{x}'}{\partial t'^2} \quad (3.21)$$

### 3.5.3 Force and pressure

By a similar approach, it can be shown that the scaling factor of the force is  $\tilde{F} \equiv \tilde{\epsilon} / \tilde{\sigma}$  so that the dimensionless force and pressure can be written as:

$$F' = \frac{\epsilon'}{\sigma'}, \quad (3.22)$$

and

$$P' = \frac{\epsilon'}{\sigma'^3}. \quad (3.23)$$

### 3.5.4 Kinetic and potential energy

$$K = \tilde{\epsilon} E'_{kin} = \frac{1}{2} m' \tilde{\epsilon} \sum_i \left( \frac{\partial^2 \tilde{x}'_i}{\partial t'^2} \right)^2, \quad (3.24)$$

and

$$E_{pot} = \tilde{\epsilon} E'_{pot}. \quad (3.25)$$

### 3.5.5 Temperature

$$T = \frac{\tilde{\epsilon}}{k_B}. \quad (3.26)$$

Therefore, for all the simulation variables a scaling factor can easily be found based Table 3.1.

### 3.5.6 Instantaneous and internal pressure

As suggested above, the instantaneous pressure,  $P_{int}$  of a system of particles such as the canonical ensemble has a kinetic energy term (the ergodic component) and a force term. In other words,

$$P_{int} = \frac{1}{3V} \left( \sum_i^N m_i \dot{\tilde{r}}_i^2 + \sum_i^N \tilde{F}_i \cdot \tilde{r}_i \right), \quad (3.27)$$

where  $V$  is the system volume. An averaging of  $P_{int}$  over time gives the internal pressure of the system. A phase transition is usually indicated by a number of system parameters changing dramatically, for instance the system pressure and the rate of diffusion.

### 3.6 Setting simulation time scale

For a given system, the time span covered by a simulation cannot be chosen arbitrarily. Clearly, if any time-based integration is to be achieved correctly, then the time interval  $\delta t$  must be defined appropriately. The energetics of a system directly affects transport parameters such as particle velocity, average particle-particle interaction time, and so on. Therefore, any time definition must necessarily start by a consideration of the energetics. Section 3.5 summarizes the important concept of dimensionless variables that allow a realistic simulation of the system to be conducted. Also, the results of any such simulation are then interpreted within the context of the dimensionless variables. For a system, such as the canonical ensemble, the following example shows how the time scale can be determined.

**Example 1** *Consider an fcc array of copper atoms ( $m=63.5\text{g/mol}$ ) with lattice parameter  $\approx 3.6\text{ nm}$  at room temperature. Suggest a simulation time scale that would be suitable to evaluate the diffusion of copper atoms from the cluster when the time increases.*

Solution: Using these parameters as reference values implies that  $\tilde{m}=1.054\times 10^{-25}\text{ kg}$  (mass of each copper atom). Similarly,  $\tilde{\sigma}=3.61\times 10^{-10}\text{ m}$ . The system is thermodynamic i.e. energetics are thermally dependent. Using the equipartition theorem while assuming only three degrees of freedom without any rotation gives:

$$\tilde{\epsilon} = \frac{3}{2}k_B T \approx \frac{3}{2} \times 1.38 \times 10^{-23} \text{ J/K} \times 300 \text{ K} \approx 6.21 \times 10^{-21} \text{ J}.$$

Application of Equation (3.19) then gives the time scale-factor:

$$\tilde{\alpha} = \sqrt{\frac{\tilde{\sigma}^2 \tilde{m}}{\tilde{\epsilon}}} = \sqrt{\frac{(3.61 \times 10^{-10})^2 \times 1.054 \times 10^{-25}}{6.21 \times 10^{-21}}} \approx 1.49 \times 10^{-12} \text{s}.$$

This scale-factor becomes the basis of examining the time evolution of the array of copper atoms. This could be interpreted in many different ways, for instance as the expected time period for a ‘complete oscillation’ of the atom about its rest position. Times shorter than this may then completely describe the time instants within a single oscillation. For instance, one could require integrations over  $N = 1000$  time steps, hence  $\delta t = \tilde{\alpha}/N \approx 0.00149$  picoseconds. Similarly, multiples of this can be used to gauge the effect of several oscillations of a given atom. For instance, one could require evaluating the system over the next 1.49 nanoseconds, which represents 1000 times the scale-factor. In which case, this could imply  $10^6$  of the earlier time intervals and be used to study the long term effects of coupled atomic oscillations in the matrix such as during phonon coupling in thermal conduction. However, it is apparent that the choice of  $\delta t$  directly impacts simulation time and accuracy. Too short and the computation takes time; too large and the accuracy suffers, requiring a careful trade-off between simulation time and accuracy.

### 3.7 Evaluation of diffusion

Diffusion is parameter of particle motion that is defined as the mean standard deviation of particle positions. Diffusion is important because significant changes in its value indicates a transition between gas to liquid to solid phases. Also, abrupt changes in system pressure are associated with a phase change (cf. Section 3.5.6). With a knowledge of the time basis and particle positions at time  $t_0$  and at a later time  $t$  allows an estimate of diffusion according to

$$\text{var}(t) = \frac{1}{N} \sum_{i=1}^N \left\| \bar{r}_i(t) - \bar{r}_i(t_0) \right\|^2 \quad (3.28)$$

This equation shows that the calculated diffusion is zero at time  $t=t_0$  and initially increases rapidly. To gauge the diffusion correctly, the calculation has to be restarted at regularly intervals.

## 3.8 Specifying the particle array

A particle is an instance of a well-defined structure that is loaded from a disk file as an space delimited ASCII text file and maintained in a RAM location. The structure definition includes the initial particle boundary conditions of position and velocity, alongside the particle mass. Listing 3.1 shows both the particle data structure definition and a specific particle instance called p. Each particle takes 72 bytes. Therefore, a macroscopic system comprising  $10^5$  particles requires 7 MB of RAM. This is only a small fraction of the RAM on a typical system i.e. 8 GB (8192 Mb). Therefore, this structure definition technically allows the software to handle millions of particles. However, as the foregoing discussion shows, the need to boost computational speed becomes paramount.

```
1 typedef struct {
2     double m;
3     double x[DIM];
4     double v[DIM];
5     double F[DIM];
6     double F_old[DIM];
7 } Particle; // 72 bytes
8 // ParticleList structure
9 typedef struct ParticleList {
10     Particle p;
11     struct ParticleList *next;
12 } ParticleList;
13
14 Particle *p;
```

```

15 double r_ij ;
16 for (d=0; d<3; d++) r_ij += sqr(p[j].x[d]-p[i].x[d]) ;
17 r_ij = pow(r,0.5) ; // calculate 3D distance

```

Listing 3.1 Definition of the particle structure used in all simulations.

Example 2 below is based on a simple arrangement of three copper atoms in a known, room-temperature stable matrix as shown in Figure 3.2. Listing 3.1 also shows how the 3D inter-particle distance  $r_{ij}$  i.e. Equation (3.14), is calculated for all MD processes. Stating the condition:  $r_{ij} > r_{cut}$ , where  $r_{cut}$  is a cut-off radius within this snippet ignores interactions beyond  $r_{cut}$  and can improve the speed of calculations.

**Example 2** *Doye & Wales have investigated global minima in transition metal SC clusters described by various (m-n) pairs i.e. (12-6), (9-6) and (10-8) for clusters up to  $N = 80$  atoms. Their published results [5] provide sufficient data to the present VSV code to illustrate its calculation of energy minima. For instance, specifying the data points in Table 3.2 for  $N = 3$  for copper with normalized energy and lattice parameters  $\varepsilon = 1$ ,  $\sigma = 1$ ,  $m = 6$ ,  $n = 9$ ,  $m_o = 63$  and  $c = 39.432$ .*

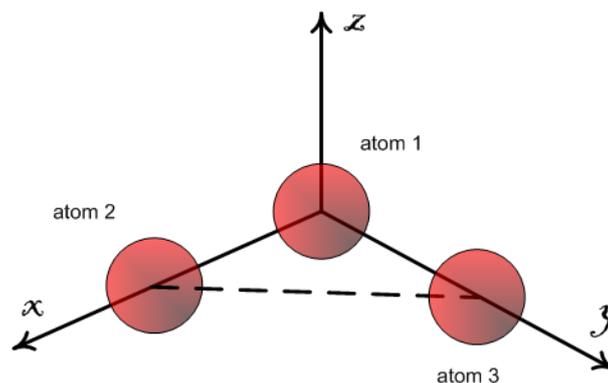


Fig. 3.2 Example of an atomic arrangement of copper atoms used in the text.

Example 2 also illustrates how VSV implements the dimensionless-quantity approach that was described in Section 3.5. That is, to conduct simulations on the energy scale of

Example 1 i.e.  $\sim 6.21 \times 10^{-21}$  J or 0.03876 eV, one could specify the energy scaling  $\varepsilon = 0.03876$  in the `sim_constants.h` file (see Listing 3.3) that is part of the VSV program. Example 2 specifically uses  $\varepsilon = 1$  to comply with the simulation conditions of Doye & Wales, particularly the temperature. By the same token for Cu,  $\sigma = 1$  implies that all lengths are divided by the lattice parameter before being passed into the VSV program. Thus, the coordinates in Table 3.2 or Listing 3.2 must be multiplied by  $3.61 \text{ \AA}$  to recover the actual atom positions in the equilibrium fcc lattice in  $\text{\AA}$ . It is evident that adjusting the simulations for different temperatures is possible by using a different energy scaling. In the subsequent chapters, where VSV is applied to solve actual physical problems, the energy scaling  $\varepsilon = 0.012382$  (eV) is used (see Table 4.1 and Table 6.1).

Table 3.2 Sample data for fcc copper obtained from [5].

| Atom | $m$ | $x$           | $y$           | $z$           | $v_x$ | $v_y$ | $v_z$ |
|------|-----|---------------|---------------|---------------|-------|-------|-------|
| 1    | 63  | -0.5264206369 | 0.1019469049  | 0.0252070114  | 0     | 0     | 0     |
| 2    | 63  | -0.8377547075 | -0.1828574677 | -0.4214018482 | 0     | 0     | 0     |
| 3    | 63  | -0.2345929501 | -0.0832085752 | -0.4827850722 | 0     | 0     | 0     |

In the method of Doye and Wales, structural identifications of metallic atomic clusters based on experimental data are improved using a Monte-Carlo technique [5]. They deform an arbitrarily constructed hypersurface of the atomic clusters for up to 80 atoms. By searching for a global potential energy minimum they are able to suggest the most stable structure of the collection of atoms. A difficulty of their method, hence the need for the globalized search, is that the global minima of adjacent surfaces are not necessarily related and therefore a newer local surface is needed for the atoms on the two surfaces. Therefore, the difficulty with the method is a complexity of  $\approx \mathcal{O}(N^2)$ . In contrast, the present method is deterministic, has a complexity of  $\approx \mathcal{O}(N \log N)$  and relies on the complete solution of the potential energy of the atomic cluster as is. Also, VSV allows the investigation of defects by arbitrary vacancy creation, atom embedding and other methods of investigation. Thus, VSV code makes no assumptions about the stability or structure of the cluster, and consequently can be applied to

a much larger number of atoms and many different boundary conditions, such as vacancies, cracks, temperature, and so on. In the case of three atoms of copper that form a known, stable structure shown in Figure 3.2, the VSV software calculates the potential energies, lists the inter-particle separations and other parameters.

Table 3.3 shows the output of the global minima calculation for the stable 3-atom copper cluster. The results shown pertain to the total potential energy ( $U$ ) of the stable 3-atom cluster calculated using Equation (3.11) under an energy scaling  $\varepsilon = 1$ . While Table 3.3 only compares 3-atom clusters with those in Doye and Wales [5], we verify their outputs for all their 80 atoms with no noticeable increase in calculation time.

Table 3.3 Program output for Table 3.2 input data.

| $N$ | 12-6       | 9-6       | 10-8      |
|-----|------------|-----------|-----------|
| 3   | -1704.6905 | -480.8560 | -633.7771 |

The VSV code calculates the energies in the system and evaluates the forces using the foregoing formulations. The basic input to the VSV code is a particle specified as a space-delimited line of text in an ASCII text file (`data.txt`) in the space-delimited 3-dimensional format specifying mass, initial positions ( $x,y,z$ ) and initial velocities ( $v_x, v_y, v_z$ ), i.e.

```
mass  x_pos  y_pos  z_pos  v_x  v_y  v_z
```

The code enumerates the file contents to determine the number of particles,  $N$ , and builds a RAM array of particles to speed up computations. The above particle data would then appear in `data.txt` as:

```
1 63  -0.5264206369  0.1019469049  0.0252070114  0  0  0
2 63  -0.8377547075  -0.1828574677  -0.4214018482  0  0  0
3 63  -0.2345929501  -0.0832085752  -0.4827850722  0  0  0
```

Listing 3.2 Sample data file for a given VSV simulation.

Listing 3.2 specifies that all three particles are at rest. However, a dimensionless initial velocity can be introduced to study the effect of an atom moving in a specific direction within

the lattice. More importantly, the velocity boundary conditions also specify the temperature through modification of the velocity algorithm. The determination of temperature is discussed in greater detail in Chapter 4. This could be done to study thermodynamic properties such as phonon stimulation and coupling, which in effect specifies heat conduction, lattice expansion, and other effects. Chapter 4 also presents the use of VSV code to determine the energetics of an ensemble in more detail.

The simulation constants are defined in a separate ASCII file `sim_constants.h` in terms of scaled energy  $\epsilon$ , lattice parameter and the exponents  $m$  and  $n$  of the LJ pairwise potential. The contents of this file are

```
1 // define the simulation constants
2 #define eps      1.0 // an energy scaling parameter
3 #define lat_const 3.61 // lattice constant of copper, in angstrom
4 #define cn      39.432 // density of state constant
5 #define nint    9 // potential function steepness
6 #define mint    6 // potential function range
7 // define macros for index mapping of cells in 3 dimensions
8 #define index(ic,nc) ((ic)[0] + (nc)[0]*((ic)[1] + (nc)[1]*(ic)[2]))
```

Listing 3.3 Typical definition of the constants for a given VSV simulation.

### 3.8.1 Software functionalities

The VSV program implements both partitioned domain or linked cell method (LCM) and unpartitioned domain simulations. In LCM the ensemble volume is subdivided into 27 partitions, which are then assigned to the available processor/core set. This set is interrogated from the operating system or cluster manager and stored as  $S=\{P, C\}$ . For the unpartitioned domain simulation, which is useful for simulations with fewer particles, the total system volume is used. Hence the user of the VSV can choose the approach to use in a given simulation. Figure 3.3 shows the flowchart of the simulation program.

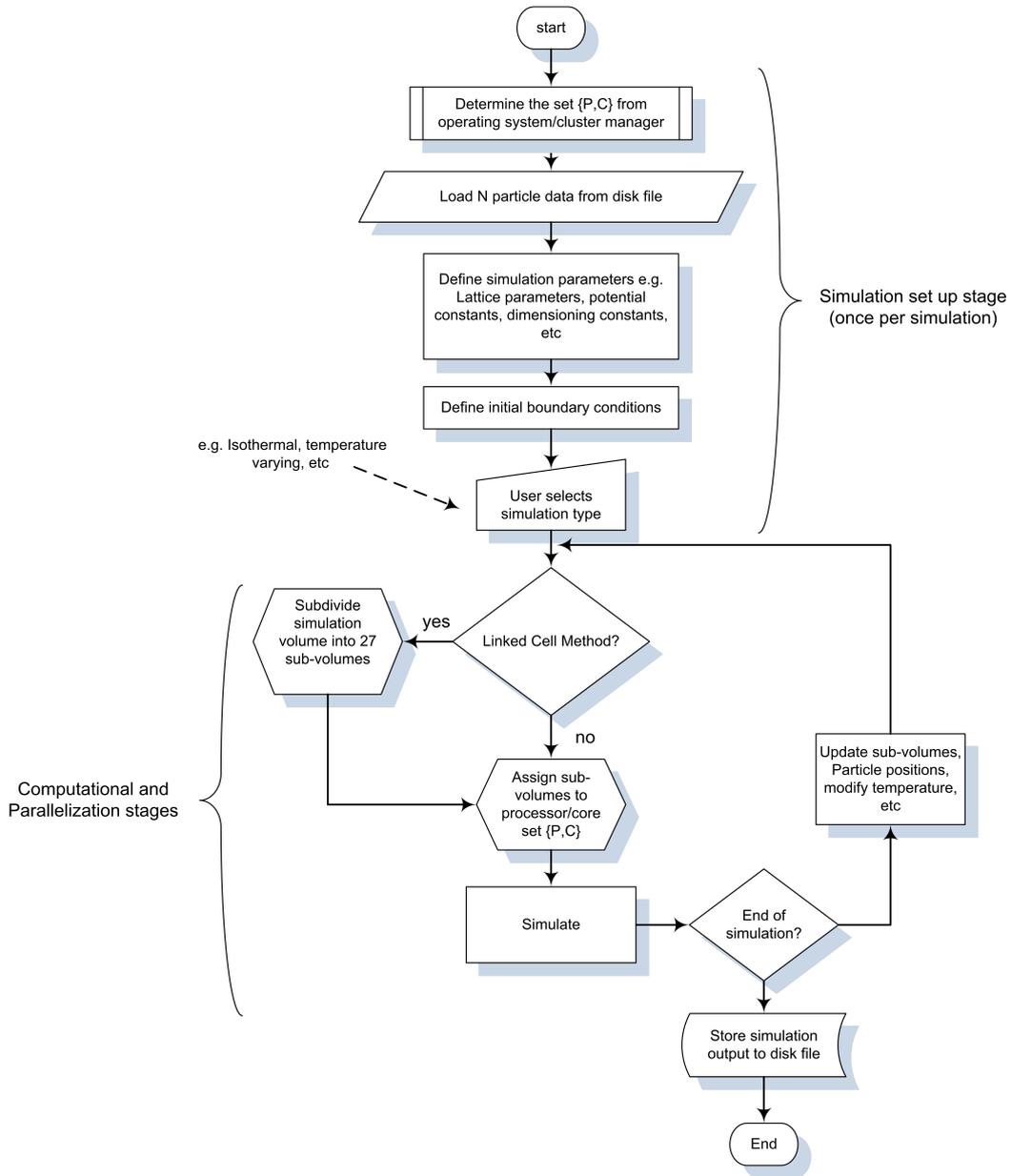


Fig. 3.3 A flowchart representing the VSV simulation program.

Figure 3.8 shows the main functionalities of the software in pictorial form. Further information about the most current build of the program code and user manuals is available for download under the GNU General Public License v3.0 [6]. The code has evolved to include the calculation of ergodics such as vacancy energy, cohesion energy, deterministic thermostats and Monte-Carlo thermostats.

Figure 3.4 shows the perturb-evaluate approach that is used within the developed VSV code. It involves determining the energy change associated with a given perturbation of a system of atoms. The atoms are first allowed to relax into a configuration of minimum potential energy,  $U_0$ . The perturbation is then applied and the system is again allowed to relax to a new steady state of minimum potential energy,  $U_F$ . The difference in potential energy,  $\Delta U = |U_F - U_0|$  is then the energy associated with the perturbation. A perturbation can be any adjustment of a particle parameter such as mass, position and velocity. Examples of specific perturbations that can be applied are:

- mass:- a more or less massive atom substitute that also forms fcc structure allows the evaluation of impurities in the matrix i.e. doping of a copper matrix with gamma-phase iron ( $\gamma$ -Fe).
- position:- removal of an atom from the matrix to a point outside the matrix can simulate a Schottky defect, a Frenkel-pair, vacuum formation, cohesive energies and so on. The term “outside” is relatively defined, for instance, it can be tens of atomic radii from the nearest or “surface”. Similarly, a point at infinity ( $r \rightarrow \infty$ ) could then be several hundreds of atomic radii away.
- velocity:- by scaling particle velocities, the effect of temperature on the matrix can be evaluated. Chapter 4 provides an indepth discussion of how this is achieved in VSV.

Note: Since a simulation is only as good as how it is applied and how the results evaluated, many other parameters can be calculated by invoking the various defined functions in the VSV toolkit. For instance, the dependence of the total potential energy of the system on the particle path can be evaluated by moving a particle along a specific Miller direction in consecutive potential energy calculations.

Figure 3.5 shows the different directions of fcc copper lattice containing 1583 atoms. These views also represent atom arrangements that were used directly to investigate the

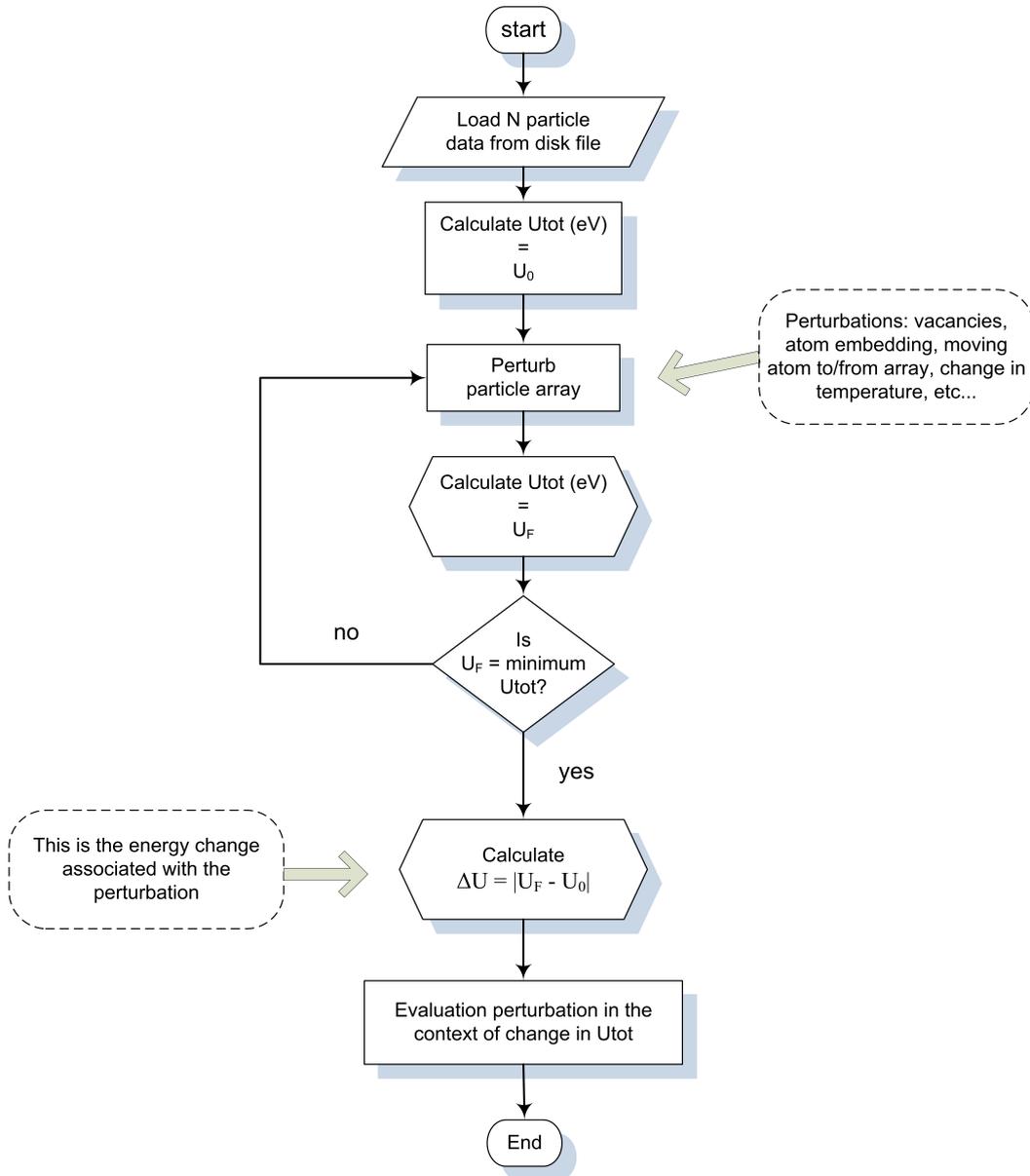


Fig. 3.4 The perturb-evaluate approach used in simulation within the VSV program.

perturbations of mass, position and velocity (temperature) of the lattice. The normal of the viewing plane intersects the page and aligns with the coordinate axis as shown.

Figure 3.6 shows the definition of a surface vacancy by the removal of an atom from a position on the surface of an ensemble of 1583 atoms on the (001) plane. The removed atom can be placed at a point at “infinity”, i.e.  $\vec{r} \rightarrow (0, 0.5, 100)\text{\AA}$ .

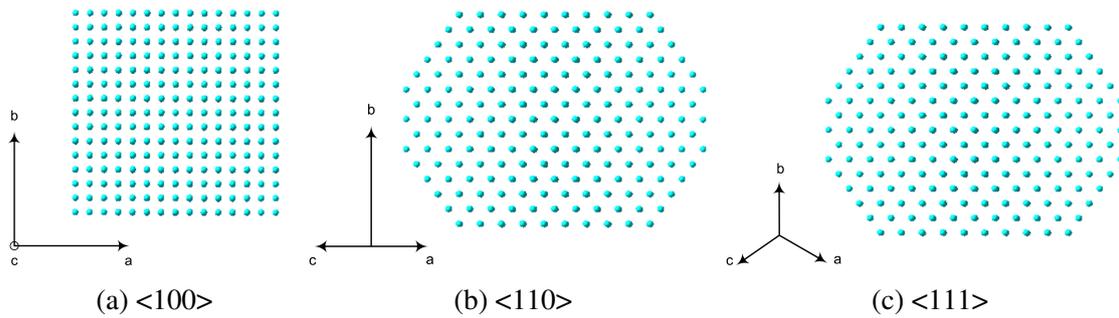


Fig. 3.5 Bird's eye, un-projected views from specific equivalent Miller index directions.

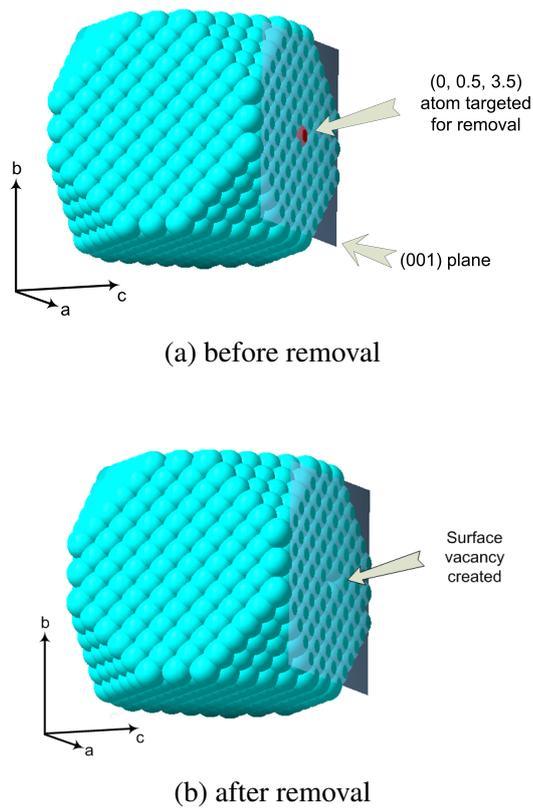
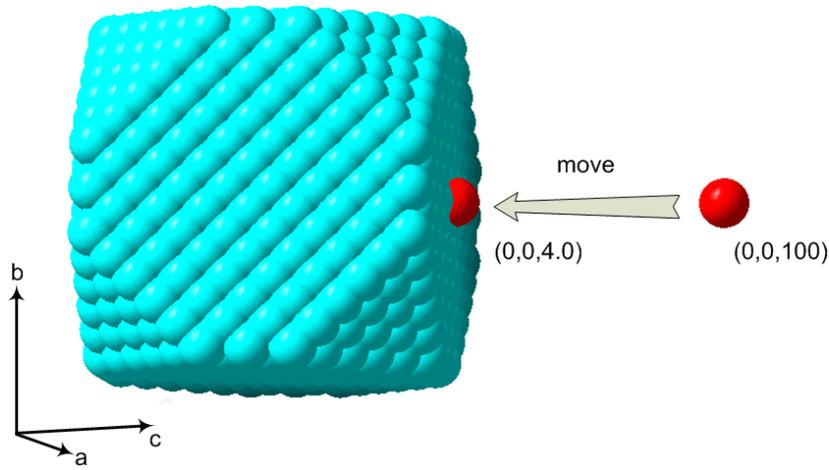
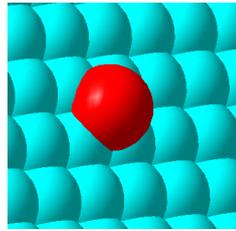


Fig. 3.6 Views of (001) plane showing vacancy creation by surface atom removal at (0, 0.5, 3.5) Å



(a) embedding



(b) close-up view

Fig. 3.7 Views of (001) plane showing the embedding of a copper atom from  $(0, 0, 100)$  Å to  $(0, 0, 4.0)$  Å.

Figure 3.7 shows the process of embedding an fcc atom onto the (001) surface from a point at infinity ( $\vec{r} \rightarrow (0, 0.5, 100)$  Å). The move arrow shows the embedding direction. This move direction defines the embedding path, which has been shown to affect the calculated  $\Delta U$ . The number of atoms in the matrix is 1583, hence the total after embedding is  $N=(1583+1)$ . The software reported in [3] calculates the average cohesive energy,  $E_{coh}$ , as the difference between the total potential energy of the perfect, undisturbed fcc lattice and when all  $N$  atoms are “blown apart” to infinite interatomic separation i.e.

$$E_{coh} = \frac{U_{apart,tot} - U_{perfect}}{N}. \quad (3.29)$$

### 3.8.2 Main code snippets

All included functions (func) are declared in the form

```
return_type func(arrangement *particle, types other_variables)
```

where arrangement can be either particle, particle list or cell array, the latter two being dedicated to the LCM method. The other\_variables can be summation indices, and so on. This approach allows single to entire particle collections to be passed back and forth as memory blocks.

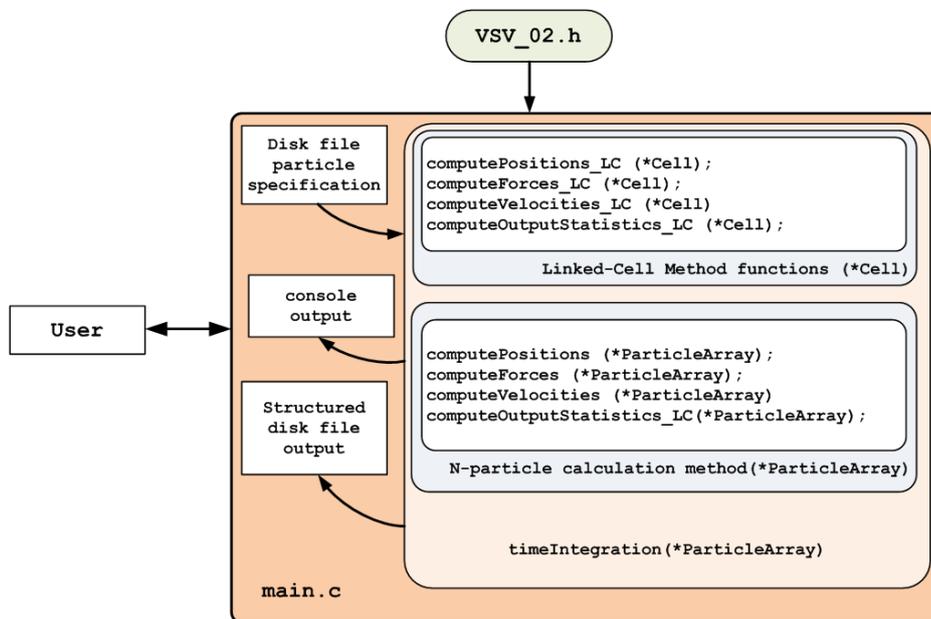


Fig. 3.8 Pictorial representation of the VSV program.

The next chapter presents the application of the VSV program to calculate two aspects of a cluster of 30,261 atoms. First, the energetics of the cluster are calculated. Within the VSV program, a given simulation is considered to be complete when  $F_i$ , which is the force acting on the  $i$ -th particle, has been determined to a specific degree of accuracy. Second, the setting and control of temperature of the ensemble is presented.

# References

- [1] Finnis MW, Sinclair JE. Long-range Finnis-Sinclair potentials. *Philosophical Magazine*, 50:45, 1984.
- [2] Sutton AP, Chen J. Long-range Finnis-Sinclair potentials. *Philosophical Magazine*, 63(1):139–156, 1990.
- [3] Ocaya R, Terblans JJ. C-language package for standalone embedded atom method molecular dynamics simulations of fcc structures. *SoftwareX*, 5:107–111, 2016. doi: 10.1016/j.softx.2016.05.005.
- [4] Verlet L. Computer “experiments” on classical fluids. I. thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159:98–103, 1984.
- [5] Doye JPK, Wales DJ. Global minima for transition metal clusters described by the Sutton-Chen potentials. *New J. Chem.*, pages 733–744, 1998.
- [6] Ocaya R, Terblans JJ. Temperature specification in atomistic molecular dynamics and its impact on simulation efficacy. *J. Phys.: Conf. Ser.*, 905(012031), 2017. doi: 10.1088/1742-6596/905/1/012031.

# Chapter 4

## Performance of the VSV software

### 4.1 Introduction

The performance of VSV code in simulations of fcc atomic structures of metallic atoms can be evaluated within two broad contexts, namely:

1. temporal - where the effects of parallelized versus unparallelized code on the same standalone platform are compared to establish any speedups,
2. spatial accuracy - where the results of actual computation of a number of known parameters for a specific type of atom, such as copper in this case, are calculated against known, empirical values in the literature.

To evaluate the performances meaningfully, a method to collect the results was developed. The reader will note that the primary focus of the present chapter is to establish any computational performance enhancements in the calculation of typical fcc crystals as early as possible during the development of VSV. Thus, the emphasis is on speedups rather than on the mathematical descriptions and significance of the calculated parameters themselves. This is reserved for the subsequent chapters. For speedup comparison, measurements of execution

time of code blocks were done using system timer functions. The following sections briefly discuss the general approach.

## 4.2 Timing functions

The function `clock()` returns the total time the operating system spent running a block of code with clear entry and exit points. However, it is not very useful for parallel threads since it sums up the running times of all threads. For instance, if four identical (parallel) threads are spawned and each takes 1 ms, then `clock()` might return 4 ms instead of 1 ms, implying incorrectly that serialization rather than parallelization is at play. Also, since thread management is automated and independent of the user, embedding `clock()` at the start of a thread is not possible. Other functions, such as `omp_get_wtime()` or `getrusage()` may then be used instead. Since the timing in the present work is only in the overall speedup, all the timings were done using `clock()`, using the flowchart shown in Figure 4.1.

```
1 // Define the entry point for the console application.
2 //
3 #include "stdafx.h"
4 #include <math.h>
5 #include <time.h>
6 #include <stdio.h>
7 #pragma omp parallel num_threads(10) for
8 int _tmain(int argc, _TCHAR* argv[]) {
9     double i;
10    double sum=0;
11    clock_t VSVstart = clock();
12    // this for loop is parallelized
13    for (i=0; i<1000;i++) {sum+=sqrt(i);
14    printf("\n Sum = %f", sum);}
15    clock_t VSVend = clock();
```

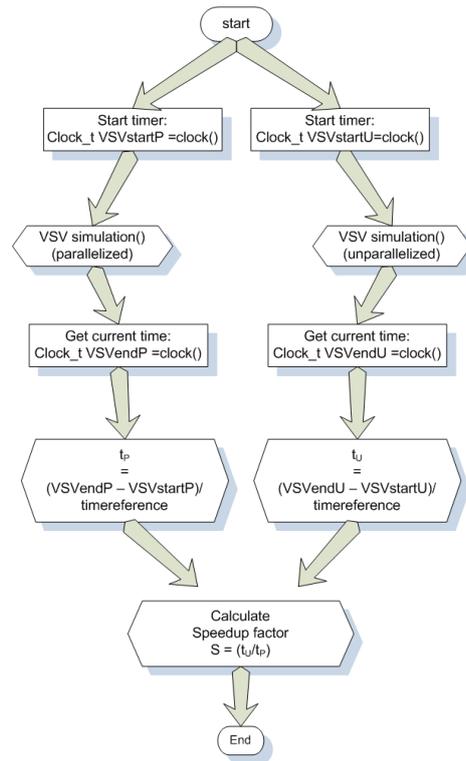


Fig. 4.1 The simple timing approach used for speedup evaluation.

```

16 printf("\n elapsed time = %f\n", (VSVend-VSVstart)/CLOCKS_PER_SEC);
17 printf("\n clocks per second = %d\n", CLOCKS_PER_SEC);
18 scanf("%f",&i);
19 return 0;
20 }
  
```

Listing 4.1 Actual Windows C code for block timing of parallelized loops.

The force-fields were derived from the embedded atom model adaptation by Sutton-Chen of the Finnis-Sinclair potential. To illustrate the speed up, we calculated the equilibrium lattice parameter ( $a$ ), the vacancy formation energy of copper, where an atom is removed for just below the surface of the bulk structure and taken to a point several hundred atomic radii away from the surface (to simulate a point at ‘infinity’), with a cut-off distance of ten lattice parameters. Similarly, we calculate the extraction energy [1, 2]. The bulk array consists of 30,261 (+1) atoms, and computation time is measured using system time functions that are

outside the particle set up, the equilibration steps and the final data output steps. The same timer functions are used for the non-parallelized iterations as well as the parallelized code. The configured hardware is the Dell Optiplex 3010 mentioned above.

### 4.3 Results of energetics and thermostat comparisons

The isotherm implementations using both the deterministic model and the Monte-Carlo method were taken to be at 400K relative to a bath temperature of 300K. The two thermostats were used to calculate under constant volume (isochoric) the following:

- cohesive energy,  $E_{ch}$ ,
- vacancy formation,  $E_v$ ,
- Schottky formation energy,  $E_{sch}$  (cf. Figure 5.1a),
- infinity-to-surface atom capture energy,  $E_{inf}$ ,
- extraction energy,  $E_{ext}$ , and
- equilibrium lattice constant,  $a_{eq}$ .

Table 4.1 lists the input parameters for the simulation of copper. All adatom effects were simulated under the assumption that they occurred relative to planes equivalent to the (100) plane.

Table 4.1 SC simulation parameters for fcc copper.

| $\varepsilon$ (eV) | $a$ (Å) | $c$    | $n$ | $m$ | $N$                |
|--------------------|---------|--------|-----|-----|--------------------|
| 0.012382           | 3.61    | 39.432 | 9   | 6   | 30,261 (+1 adatom) |

Table 4.2 shows the results of the two thermostats. The point at infinity is taken approximately 100 lattice parameters away from the bulk of the array, since the atoms are assumed

to be in a closed box of fixed volume. The results shown in Table 4.2 were compared with Table 4.2 Comparison of energies calculated using deterministic (A) versus MC (B) thermostats on Cu<sup>(100)</sup> at 400K.

| Method | $a_{eq}$ (Å) | $E_v$     | $E_{ext}$ | $E_{coh}$<br>(eV) | $E_{sch}$ | $E_{inf}$ |
|--------|--------------|-----------|-----------|-------------------|-----------|-----------|
| A      | 3.63±0.01    | 1.31±0.02 | 4.39±0.01 | 1.28±0.01         | 4.41±0.01 | 2.96±0.01 |
| B      | 3.74±0.02    | 1.66±0.30 | 5.84±0.62 | 2.17±0.60         | 5.98±0.60 | 3.50±0.50 |

literature values, where abundant empirical data exists in the literature at 400K for the deterministic approach in Method A [3–6], than for the MC approach, Method B. The MC method requires aggregation over several steps per move to give a result obeying the canonical distribution, and the actual reported results in Table 4.2 are time-based averages of several computations per move, with a move acceptance probability of around 30%. The move perturbations are selected randomly through the Markov-chain approach, with a probability dependent on  $\Delta x^N$ , which represent all possible generated combinations due to perturbing inter-particle separation,  $x$ . The difficulty with the MC approach is therefore appreciable for larger ensembles owing to the total size of possible configurations. This accounts for the lower MC method accuracies depicted in the Table 4.2.

## 4.4 Timing performance

Table 4.3 Determined speed-up in energy calculations based on deterministic (A) versus MC (B) thermostats on Cu<sup>(100)</sup> at 400K.

| Method           | $a_{eq}$<br>(Å) | $E_v$<br>(eV) | $E_{ext}$<br>(eV) | Speed up |
|------------------|-----------------|---------------|-------------------|----------|
| EPT              | 0.362±0.01      | 1.31          | 4.39              | 1.8      |
| Monte Carlo (5%) | 0.371±0.02      | 1.66          | 5.84              | 2.2      |

Table 4.3 summarizes the time-factors of the two methods observed on 30,261 atoms. Without parallelization, the run times in a calculation of  $E_v$  on an Ubuntu Linux terminal were 6.70 hours and 5.21 hours for the deterministic and Monte Carlo simulations respectively. The maximum parallelized speedups are shown in Table 1. The results show a speed up of 1.8 on the wall-clock time for 5% accuracy on the lattice parameter with a reduced cut-off radius of 10 radii. The Monte-Carlo method exhibited higher speed up, with the lower acceptance rate, much higher than observed for the deterministic, MD based simulation.

The program is actively evolving but a published version of the software can be obtained from a permanent repository [7, 8].

## 4.5 Investigation of real clusters

The chapters that follow describe actual applications of the VSV software to investigate actual atomic clusters to gain insight into their behaviors and to advance the knowledge of such clusters. Specifically, Chapter 5 clarifies the temperature specification for such systems, Chapter 6 studies low temperature particle motions and argues the case for diffusion. Chapter 7 shows that in bond-length oscillations are detectable in within the SC model in the VSV implementation.

# References

- [1] Ocaya R, Terblans JJ. C-language package for standalone embedded atom method molecular dynamics simulations of fcc structures. *SoftwareX*, 5:107–111, 2016. doi: 10.1016/j.softx.2016.05.005.
- [2] R O Ocaya and J J Terblans. Coding considerations for standalone molecular dynamics simulations of atomistic structures. *Journal of Physics: Conference Series*, 905(1): 012018, 2017. doi: 10.1088/1742-6596/905/1/012018. URL <http://stacks.iop.org/1742-6596/905/i=1/a=012018>.
- [3] Terblans JJ. Calculating the bulk vacancy formation energy (Ev) for a Schottky defect in a perfect Cu(111), Cu(100) and a Cu(110) single crystal. *Surf. Interface Anal.*, 33: 767–70, 2002. doi: 10.1002/sia.1451.
- [4] Kraftmakher Y. Equilibrium vacancies and thermophysical properties of metals. *Phys. Rep.*, 299(2-3):79–188, 1998. doi: 10.1016/S0370-1573(97)00082-3.
- [5] Fluss MJ, Smedskjaer LC, Siegel RW, Legnini DG, Chason MK. in *Hasiguti RR, Fujiwara K (Ed.) Proceeding of the Fifth International Conference on Positron Annihilation, April 8–11, 1979*, volume 97. Japan Institute of Metals, Lake Yamanaka, Japan, 1979.
- [6] Tritshäuser W, McGervey JD. Monovacancy formation energy in copper, silver, and gold by positron annihilation. *Appl. Phys.*, 6(2):177–180, 1975. doi: 10.1007/BF00883748.

- 
- [7] Ocaya RO, Terblans JJ. Velocity-Verlet-Störmer software, VSV. *Software in a repository*, 2016. URL <https://github.com/ElsevierSoftwareX/SOFTX-D-15-00054>.
- [8] Ocaya R, Terblans JJ. Addressing the challenges of standalone multi-core simulations in molecular dynamics. in *P. Ramasami (ed.), Computational Sciences, De Gruyter*, pages 1–21, 2017. doi: 10.1515/9783110467215-001.

# Chapter 5

## Specifying system ergodicity in VSV

### 5.1 Introduction

Temperature is a vital thermodynamical function for physical systems. In the present case, we investigate systems that are said to be *ergodic*. These are systems that, when their time-averaged dynamical properties are compared with their spatially-averaged phase space trajectories, exhibit the same behaviour. Theoretical and computational developments in many fields, such as condensed matter physics, chemistry, material science, molecular biology, nanotechnology and many others, implicitly and crucially rely on the temperature specification. It is therefore expected that temperature-based simulations of materials will grow in prominence regardless of the computing paradigm used. The knowledge of temperature facilitates the understanding of the system's ergodicity, leading to clarification of system state and stability. System state can be specified using the concept of phase space in different ensembles. In the canonical ensemble, mentioned in Chapter 3, this is done through  $N$ ,  $V$  and  $T$ . The system pressure ( $P$ ) is implicitly determinable.

This chapter presents a secondary application of the developed program to specify temperature through isotherms and discusses the apparent variability of temperature modeling within its various, current formalisms in MD simulations. This variability may be expected

to affect the overall energetics, dynamics and structural evolution in terms of differences in calculated outputs. This fundamental question is not openly discussed in the heuristic commercial simulation programs of the earlier chapters because the occasion never arises to do so. This is an additional, clear advantage of the present approach. The chapter compares two temperature specifications reported in the literature. Using the first approach, a thermostat is defined deterministically at 400K relative to a heat bath at 300K using a modification of the standard Newtonian method. Then, using the second approach, a Monte-Carlo method is employed, assuming stochastic displacements of particle positions. For a meaningful comparison of the two approaches, the thermostatic vacancy formation and cohesion energies, equilibrium lattice constant for fcc copper are calculated.

## 5.2 Thermostat definitions

A thermostat may be thought of as a system at constant temperature. In our particular context, it represents the canonical ensemble  $(N, V, T)$  at given temperature  $T$ . This is not to say that  $T$  will remain the same throughout the existence of the system, but that for each calculated stable phase of the system the temperature remains at the specified value. In general, thermostats are defined in two ways.

1. deterministically, using the energy equipartition theorem (EPT), and
2. stochastically, for instance using a Monte-Carlo (MC) approach.

### 5.2.1 Energy equipartition methods

A time instant  $\delta t$  of simulation deterministically defines the evolution of length and energy parameters. This means that given a particular input, the application of the iterative algorithm

will produce the same result at time  $\delta t$ . System energetics are conveyed by the Hamiltonian,

$$\mathcal{H}(\bar{x}, \bar{p}) = \mathcal{U}(\bar{x}) + \mathcal{K}(\bar{p}),$$

where  $\mathcal{U}$  and  $\mathcal{K}$  are total kinetic and potential energy respectively,  $\bar{x}$  is displacement and  $\bar{p}$  is momentum. Instantaneous temperature  $T(t)$  is defined in terms of the instantaneous internal kinetic energy  $\varepsilon$ . Then, macroscopic system temperature equals averaged instantaneous temperature, i.e.  $T = \langle T(t) \rangle$  [1]. In the thermostatic approach variance in the Hamiltonian, as might arise from computational error accumulation, is tolerable because the thermostat is referenced to a heat bath that tightly constrains temperature variations about the equilibrium value. At equilibrium, the standard deviation  $\sigma(\mathcal{H})$  is related to isochoric heat capacity,  $c_V$ , by

$$\sigma^2(\mathcal{H}) = \langle \mathcal{H}^2 \rangle - \langle \mathcal{H} \rangle^2 = k_B T^2 c_V, \quad (5.1)$$

where  $k_B$  is Boltzmann constant. Internal particle motion within a system of particles is generally a complex combination of translations and rotations about a center of mass which contribute three and six degrees of freedom respectively. Computational simplifications can be achieved by subtracting the degrees of freedom when stochastic forces (such as Brownian motion) and friction are absent from the system. This is done because degrees of freedom that are not mechanically coupled do not exchange energy [2, 3]. If the center of mass is at rest and the particle motions are irrotational, then

$$T = \frac{2}{3Nk_B} \varepsilon = \frac{2}{3Nk_B} \sum_{i=1}^N \frac{m_i}{2} \bar{v}_i^2. \quad (5.2)$$

Using the discrete notation  $\bar{p}_i(t) := \bar{p}_i^n$ , then in the absence of friction and stochastic forces the velocity Störmer-Verlet method [4–6] discretizes particle position and velocity as

$$\bar{x}_i^{n+1} = \bar{x}_i^n + \delta t \bar{v}_i^n + \frac{\bar{F}_i^n}{2m_i} \delta t^2, \quad \text{and} \quad \bar{v}_i^{n+1} = \bar{v}_i^n + \left( \frac{\bar{F}_i^n + \bar{F}_i^{n+1}}{2m_i} \right) \delta t. \quad (5.3)$$

Since instantaneous temperature  $T(t)$  is directly related to atomic velocities, thermostats require a mechanism to control the rate of change of particle velocities. This can be achieved either by introducing friction or by direct velocity scaling, as described below.

### 5.2.2 Langevin EPT methods

In the first method or Langevin approach [7–9], Newton’s second law is modified by introducing a stochastic force  $\bar{R}_i(t)$  and a friction coefficient  $\gamma_i(t) > 0$  into the acceleration of the  $i$ -th particle:

$$\dot{\bar{v}}_i(t) = \frac{\bar{F}_i(t)}{m_i} - \gamma_i(t) \bar{v}_i(t) + \frac{\bar{R}_i(t)}{m_i} \quad (5.4)$$

In the absence of stochasticity, the friction term loses its traditional meaning and only indicates heat flow direction relative to the infinite heat bath. In which case,  $\gamma_i(t) > 0$  implies decelerating particles and consequently loss of heat to the bath. Conversely,  $\gamma_i(t) < 0$  implies heat gain from the bath and accelerating particles. This can also be interpreted as isochoric system pressure fluctuations, since pressure is in effect the force acting per unit area of a given surface.

### 5.2.3 Velocity-scaled EPT thermostats

The second method, which is also known as velocity scaling, starts at Equation (5.2) to define a scaling factor  $\beta$  in terms of temperatures, kinetic energies and consequently velocities

[10–13] i.e.

$$\beta^2 := \frac{T'}{T} = \frac{\varepsilon'}{\varepsilon} = \left(\frac{\bar{v}'_i}{\bar{v}_i}\right)^2 \Rightarrow \bar{v}' := \beta \bar{v}. \quad (5.5)$$

After every  $\delta t$  step, the velocity is therefore reassigned a value  $\bar{v}'$  instead of  $\bar{v}$  for the new temperature to be  $T'$ . Velocity scaling using constant  $\beta$  affects temperature distribution strongly, and a damping factor  $\Gamma \in (0,1)$  can be introduced [11] s.t.  $\beta = (1 + \Gamma(T'/T - 1))^2$ . If  $\beta$  is adjusted at the end of every time step  $\delta t$ , then  $dT/dt \sim (T - T')$ , i.e. the temperature rate of change depends on the temperature difference. The method does not remove local correlations in particle motions. Setting  $\gamma_i^n(t) = \xi^n m_i$ , the non-stochastic net force is

$$m_i \frac{\bar{v}_i^{n+1} - \bar{v}_i^n}{\delta t} = \bar{F}_i^n - \xi^n m_i \bar{v}_i^n, \quad \text{for } \delta t \rightarrow 0 \quad (5.6)$$

with new velocities

$$\bar{v}_i^{n+1} = \frac{\bar{F}_i^n \delta t}{m_i} + \beta \bar{v}_i^n, \quad (5.7)$$

where the term  $\beta = (1 - \xi^n \delta t)$  describes the friction-time behavior. Using  $\bar{F}_i^n$  and  $\bar{F}_i^{n+1}$  in Equations (5.3) and (5.6) gives new position and velocity:

$$\bar{x}_i^{n+1} = \bar{x}_i^n + \beta \bar{v}_i^n \delta t + \frac{\bar{F}_i^n}{2m_i} \delta t^2, \quad \bar{v}_i^{n+1} = \frac{1}{\varphi} \left[ \beta \bar{v}_i^n + \left( \frac{\bar{F}_i^n + \bar{F}_i^{n+1}}{2m_i} \right) \delta t \right], \quad (5.8)$$

where  $\varphi = (1 + \frac{\delta t}{2} \xi^{n+1})$ . Therefore, friction amounts to scaling the velocity at the current time step first by  $\beta$ , then overall by  $1/\varphi$ . To obtain an expression for a thermostat, which is really a constant temperature state, one notes first that the kinetic energy is constant i.e.  $d\varepsilon/dt=0$ . Then, for the  $i$ -th particle

$$m_i \dot{\bar{v}}_i = \bar{F} - \xi m_i \bar{v}_i, \quad \frac{d\varepsilon}{dt} = \sum_{i=1}^N m_i \bar{v}_i \cdot \dot{\bar{v}}_i = \sum_{i=1}^N \bar{v}_i \cdot (\bar{F} - \xi m_i \bar{v}_i). \quad (5.9)$$

Writing  $\bar{F} = -\bar{\nabla}V_{\bar{x}_i}$  leads to

$$\frac{d\varepsilon}{dt} = -\left(\frac{dV}{dt} + \xi \sum_{i=1}^N m_i \bar{v}_i^2\right), \quad (5.10)$$

hence the constant temperature condition:

$$\xi = -\frac{dV/dt}{\sum_{i=1}^N m_i \bar{v}_i^2} = \frac{dV/dt}{2\varepsilon}. \quad (5.11)$$

Some thermostats introduce an extra degree of freedom in the form of heat bath temperature. This conveys the extent of system–bath coupling [12, 13]. Since  $\xi^n$  arises from friction, the rate of system kinetic energy loss is related to the friction-time behaviour:

$$\frac{d\xi}{dt} = \frac{\left(\sum_i^N m_i \bar{v}_i^2 - 3Nk_B T^{\text{bath}}\right)}{M}, \quad \text{for } M \in \mathbb{R}^+, \quad (5.12)$$

where  $M \gg 1$  implies less frictional coupling. To determine  $\bar{v}_i^{n+1}$  in Equation (5.8), the term  $\xi^{n+1}$  must be known. Many nonlinear methods [14–16] arise from discretizations of Equation (5.12), e.g. midpoint time averaging:

$$\xi^{n+1} = \xi^n + \frac{\delta t}{2M} \left( \left. \frac{d\xi}{dt} \right|^n + \left. \frac{d\xi}{dt} \right|^{n+1} \right) \quad (5.13)$$

gives one possible thermostat when  $d\xi/dt$  from Equation (5.12) is used. Simpler, more approximate discretizations are possible. In general, a Hamiltonian system may have stable integrators but the equations of motion may not be derivable. If the Hamiltonian is time-independent [17, 18] and can be written as

$$\mathcal{H}(\bar{x}, \bar{p}) = \sum_{i=1}^N \frac{\bar{p}_i^2}{2m_i \gamma^2} + V(\bar{x}_1, \dots, \bar{x}_N) + \frac{\bar{p}_\gamma^2}{2M} + 3Nk_B T^{\text{bath}} \ln \gamma, \quad (5.14)$$

then the equations of motion and phase space can be derived. The phase space may be defined in terms of intensive macroscopic variables, which are aggregates of microscopic contributions. Explicit reference to bath temperature generally avoids systematic energy drifts, implying ergodicity [19–21]. In such a system, particles cannot leave the ensemble. Therefore, assuming equal Maxwell-Boltzmann particle distribution probabilities and ergodicity *a priori* may result in an NVT (see Section 3.1.1) ensemble of microstates and momenta [17]. The probability distribution of microstates over radial and momentum space takes the form

$$\rho(\bar{x}, \bar{p}) = \frac{e^{-\mathcal{H}(\bar{x}, \bar{p})/k_B T}}{\iint e^{-\mathcal{H}(\bar{x}, \bar{p})/k_B T} d\bar{x} d\bar{p}}. \quad (5.15)$$

Putting  $\bar{p}_i = m_i \dot{\bar{x}}_i$  allows simplification in Cartesian coordinates. The particle velocities can be shown to also follow a Maxwell-Boltzmann distribution. There are other MD thermostats that are specified under various limiting conditions [17, 22, 23]. Caution is needed when comparing thermostats grouped holistically within the MD approach. This is because a particular MD thermostat implementation may facilitate determination of a particular parameter, but is not generally advantageous to all situations. For instance, Hoover-Evans [20, 24] and Haile-Gupta [23] thermostats are inherently time-reversible, whereas Andersen [10] and Berendsen [11] thermostats are not.

#### 5.2.4 Monte Carlo methods

A simpler thermostat can be defined non-deterministically by sampling relative particle positions using Monte-Carlo (MC) techniques [17, 25]. This involves testing the effort of a random displacement of a particle within the domain (i.e. box dimensions) against the change in potential energy  $\Delta\mathcal{U}(\bar{x})$  without involving kinetic energy. The move is “acceptable” if a probability  $p$  can be found such that

$$p = \min\{e^{-\beta\Delta\mathcal{U}(\bar{x})}, 1\}, \quad \text{where } \beta = 1/k_B T. \quad (5.16)$$

The microstate probability distribution  $\rho(\bar{x})$  is then

$$\rho(\bar{x}) = \frac{e^{-\beta \mathcal{U}(\bar{x})}}{\int e^{-\beta \mathcal{U}(\bar{x})} d\bar{x}}. \quad (5.17)$$

This form, where particle positions are sampled exponentially without geometrical constraints, encapsulates a thermostat in the canonical ensemble that obeys Maxwell-Boltzmann (MB) thermodynamics. The microstate probability can be presented as the acceptable accuracy for the test, e.g. 20% of final position.

### 5.2.5 Timescale and macroscopics

Timescale can be thought of as times larger than the relaxation time in inter-atomic particle collisions but smaller than the shortest experimental observation time for the system. Being a system simulation as well, the timescale is also taken as a dimensionless quantity. The deterministic approach, unlike the MC approach, is explicit in the timescale. This naturally leads to the fundamental question of whether time-correlation functions can be found in the MC approach to describe macroscopic time-based phenomena. Instantaneous particle positions depend on the time behavior and temperature spread over particles. It is therefore important to deal with temperature-time behavior in the MC method, at the very least qualitatively, in the absence of analytical expressions [10].

The average temperature per particle is taken as  $\bar{T}$  for a system that is in contact with a bath at temperature  $T^{\text{bath}}$ . Many time-based system parameters of interest, such as diffusion coefficients, are calculated as statistical averages in the Einstein formulation [26], or by time-correlation integrals in the Green-Kubo formulation [27, 28]. At constant volume  $V$ , the change in average energy is  $d\bar{E} = c_v d\bar{T}$ . By Newton's law of cooling:

$$\frac{1}{c_v} \frac{d\bar{E}}{dt} = \frac{d\bar{T}}{dt} = \alpha (T^{\text{bath}} - \bar{T}), \quad \text{where } \alpha = \frac{c_v}{\kappa V^{1/3}}. \quad (5.18)$$

The constant  $\kappa$  encapsulates temperature inhomogeneities and system shape and is estimated from the heat flow continuity and flux equations:

$$\begin{aligned}\bar{J}(\bar{x}, t) &= -\kappa \bar{\nabla} T(\bar{x}, t) \\ \frac{\partial T(\bar{x}, t)}{\partial t} &= -\frac{V}{c_v} \bar{\nabla} \cdot \bar{J}(\bar{x}, t).\end{aligned}\tag{5.19}$$

Solving Equation (5.18) gives

$$\bar{T}(t) = T_0 + [\bar{T}(0) - T^{\text{bath}}] e^{-\alpha t},\tag{5.20}$$

on a timescale such that stochasticity in  $\bar{T}(t)$  averages out. From purely theoretical considerations the choice of either a specific MD, or even the MC thermostat, is subjective. However, one expects correct dynamics with a thermostat that accommodates temperature fluctuations on the timescale.

### 5.3 Comparative simulation of EPT and MC thermostats

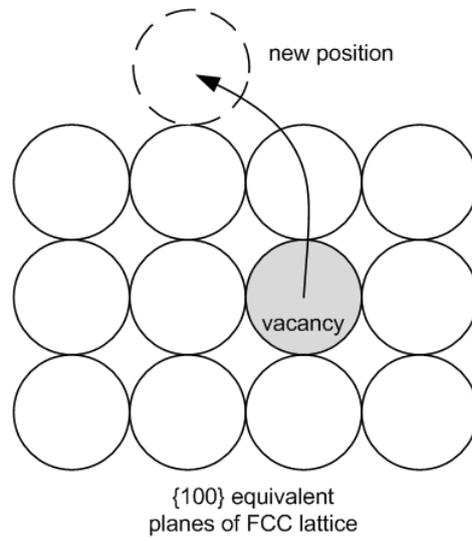
The present work has investigated the effect of the foregoing thermostat definitions, deterministic and Monte Carlo, on the equilibrium lattice constant and formation energies on an isochoric, monoatomic array of  $n=1583$  (plus one, if with a surface adatom) copper atoms in the fcc structure, and a heat bath at 300K where necessary. The chosen system for the test simulation is deliberately simpler for speedier computation and comparison of the thermostats. The deterministic approach employs the Sutton-Chen (SC) embedded atom model (EAM) form of the Finnis-Sinclair potential [29–32]. The authors have previously reported their implementation of SC code [3], which was modified for the present work to accommodate velocity scaling-based thermostats using Störmer-Verlet integration [6], as well as the MC thermostat with a 30% acceptance probability. EAM methods are not purely

deterministic but their lower computational effort for more particles and better handling of larger timescales is attractive [33]. In [31, 33, 34], the EAM Hamiltonian is given by Equation (3.10). As mentioned earlier, it takes into account the embedding energy of atom  $i$  as a function of the host electron density. For fcc metals the translation and rotation invariant total potential energy [4, 29, 35, 36] is given by Equation (3.11). The fitting constants are  $c$ ,  $m$  and  $n$ , where  $m < n$ . The force on the  $i$ -th particle is given by Equation (3.13).

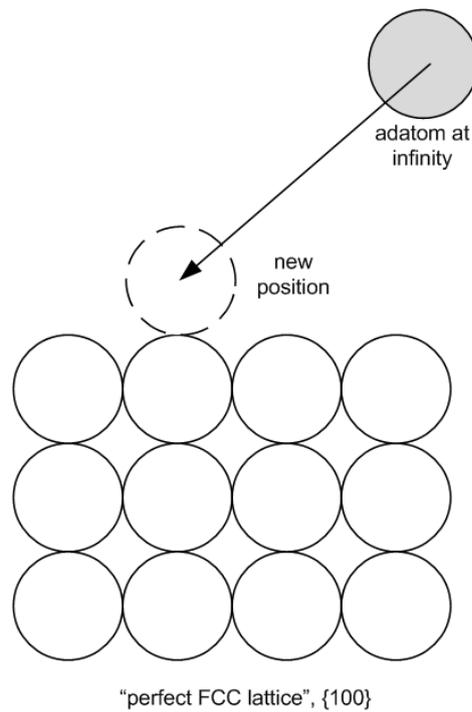
The Hamiltonian ( $\mathcal{H}$ ) is here accepted to within  $O(\delta t^2)$  discretization error, but is ideally constant [37]. Other temperature dependent effects like diffusion can be investigated through the effect of the thermostat definition on existing literature models. For instance, diffusion [38] is probabilistically modeled by

$$D = D_0 \prod P_m P_v = D_0 e^{-E_m/k_B T} e^{-E_v/k_B T}, \quad (5.21)$$

where  $P_m$  and  $P_v$  are inter-lattice point migration and vacancy availability probabilities respectively, expressed in terms of migration energy  $E_m$  and vacancy formation energy  $E_v$ . These energies can be evaluated for each thermostat  $T$ , by considering the various vacancy formation and migratory mechanisms for the perfect crystal lattice under adatom perturbation [39]. We consider only for the  $\{100\}$  equivalent planes of the lattice-point to surface Schottky defect creation shown in Fig.5.1a and from-infinity to surface migration, shown in Fig.5.1b.



(a) Schottky defect formation by surface migration.



(b) From-infinity to surface migration.

Fig. 5.1 Two migrations associated with the {100} equivalent planes of the fcc lattice.

# References

- [1] D. Chandler. *Introduction to Modern Statistical Mechanics*. Oxford University Press, New York, 1987.
- [2] Graben HW, Ray JR. Global minima for transition metal clusters described by the Sutton-Chen potentials. *Phys. Rev. A*, 43:4100–4103, 1991.
- [3] Ocaya R, Terblans JJ. C-language package for standalone embedded atom method molecular dynamics simulations of fcc structures. *SoftwareX*, 5:107–111, 2016. doi: 10.1016/j.softx.2016.05.005.
- [4] Griebel M, Knapek S, Zumbusch G et al (Eds.). Numerical simulation in molecular dynamics. *in, Texts in Computational Science and Engineering 5, Springer, Berlin, 5* (ISBN 978-3-540-68094-9), 2007.
- [5] Hockney R. The potential calculation and some applications. *Methods Comp. Phys.*, 9: 135–211, 1970.
- [6] Verlet L. Computer “experiments” on classical fluids. I. thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159:98–103, 1984.
- [7] Schneider T, Stoll E. Molecular-dynamics study of a three-dimensional one-component model for distortive phase transitions. *Phys. Rev. B*, 17:1302–1322, 1978.

- 
- [8] Wang W, Skeel RD. Analysis of a few numerical integration methods for the Langevin equation. *Mol. Phys.*, 101:2149–2156, 2003.
- [9] Berkowitz M, Morgan JD, McCammon JA. Generalized Langevin dynamics simulations with arbitrary time-dependent memory kernels. *J. Chem. Phys.*, 78:3256–3261, 1983. doi: 10.1063/1.445244.
- [10] Andersen H. Molecular dynamics simulations at constant pressure and/or temperature. *J. Chem. Phys.*, 4(72):2384–2393, 1980. doi: 10.1063/1.439486.
- [11] Berendsen H, Postma J, van Gunsteren W, Di Nola A, Haak J. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.*, 72:3684–3690, 1984. doi: 10.1063/1.448118.
- [12] Hoover W. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31:1695–1697, 1985. doi: 10.1103/PhysRevA.31.1695.
- [13] Nosé S. A molecular dynamics method for simulations in the canonical ensemble. *Mol. Phys.*, 52(101):255–268, 1984. doi: 10.1080/00268978400101201.
- [14] Frenkel D, Smit B. *Understanding molecular simulations: from algorithms to applications*. Academic Press, New York, 1996.
- [15] Gear C. *Numerical initial value problems in ordinary differential equations*. Prentice-Hall, New Jersey, 1996.
- [16] Bond S, Leimkuhler B, Laird B. The Nosé–Poincaré method for constant temperature molecular dynamics. *J. Comput. Phys.*, 151(1):114–134, 1999.
- [17] Hünenberger PH. Thermostat algorithms for molecular dynamics simulations. *Adv. Polym. Sci.*, 173:105–149, 2005. doi: 10.1007/b99427.

- [18] Allen MP, Tildesley DJ. *Computer Simulation of Liquids*. Oxford University Press, New York, 1987.
- [19] Rugh HH. Dynamical approach to temperature. *Phys. Rev. Lett.*, 78:772–774, 1997.
- [20] Evans DJ, Sarman S. Equivalence of thermostatted nonlinear responses. *Phys. Rev. E.*, 48(65):65–70, 1993.
- [21] Butler BD, Ayton G, Jepps OG, Evans DJ. Configurational temperature: Verification of Monte Carlo simulations. *J. Chem. Phys.*, 109:6519–6522, 1998. doi: JCPA6.
- [22] Woodcock LV. Isothermal molecular dynamics calculations for liquid salts. *Chem. Phys. Lett.*, 10(3):257–261, 1971. doi: 10.1016/0009-2614(71)80281-6.
- [23] Haile JM, Gupta S. Extensions of the molecular dynamics simulation method. ii. isothermal systems. *J. Chem. Phys.*, 79(6):3067–3076, 1983.
- [24] Hoover WG, Ladd AJC, Moran B. High-strain-rate plastic flow studied via nonequilibrium molecular dynamics. *Phys. Rev. Lett.*, 48:1818–1820, 1982. doi: 10.1103/PhysRevLett.48.1818.
- [25] van Gunsteren WF, Nanzer AP, Torda AE. Molecular simulation methods for generating ensembles or trajectories consistent with experimental data, in: Binder K, Ciccotti G (eds). *Monte Carlo and molecular dynamics of condensed matter systems, Proceedings of the Euroconference, SIF, Bologna, Italy*, 49:777–788, 1995.
- [26] Bose Z. Planck’s law and light quantum hypothesis. *Zeitschrift für Physik*, 26(1): 178–181, 1924. doi: 10.1007/BF01327326.
- [27] Green MS. Markoff random processes and the statistical mechanics of time-dependent phenomena. II. Irreversible processes in fluids. *J. Chem. Phys.*, 22:398–413, 1954.

- [28] Kubo R. Statistical-mechanical theory of irreversible processes. i. general theory and simple applications to magnetic and conduction problems. *J. Phys. Soc. Jpn.*, 12: 570–586, 1957.
- [29] Finnis MW, Sinclair JE. Long-range Finnis-Sinclair potentials. *Philosophical Magazine*, 50:45, 1984.
- [30] Daw MS, Foiles SM, Baskes MI. The embedded-atom method: a review of theory and applications. *Mater. Sci. Rep.*, 9(7-8):251–310, 1993. doi: 10.1016/0920-2307(93)90001-U.
- [31] Daw MS, Baskes MI. Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals. *Phys. Rev. B*, 29:6443–6453, 1984. doi: 10.1103/PhysRevB.29.6443.
- [32] Todd BD, Lynden-Bell RM. Surface and bulk properties of metals modelled with Sutton-Chen potentials. *Surface Science*, 281:191–206, 1993. doi: 10.1016/0039-6028(93)90868-K.
- [33] Chamati H, Papanicolaou NI, Mishin Y, Papaconstantopoulos DA. Embedded-atom potential for Fe and its application to self-diffusion on Fe(1 0 0). *Surface Science*, 600: 1793–1803, 2006. doi: 10.1016/j.susc.2006.02.010.
- [34] Mishin Y. in: *Yip S (Ed.), Handbook of Materials Modeling*, volume 459. Springer, The Netherlands, 2005.
- [35] Doye JPK, Wales DJ. Global minima for transition metal clusters described by the Sutton-Chen potentials. *New J. Chem.*, pages 733–744, 1998.
- [36] Swope W, Andersen H, Berens P, Wilson K. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.*, 76(1):637–649, 1982.

- 
- [37] Landau L, Lifschitz E. *Mechanics, Course of Theoretical Physics*, volume 1. Pergamon Press, Oxford, 1976.
- [38] Terblans JJ, Erasmus WJ, Viljoen EC. Orientation dependence of the surface segregation kinetics in single crystals surface and interface analysis. *Surface Interface Anal.*, 28: 70–72, 1999.
- [39] van der Walt C, Terblans JJ, Swart HC. Molecular dynamics study of the temperature dependence and surface orientation dependence of the calculated vacancy formation energies of Al, Ni, Cu, Pd, Ag, and Pt. *Comput. Mater. Sci.*, 83:70–77, 2014. doi: 10.1016/j.commatsci.2013.10.039.

# Chapter 6

## Low temperature diffusion and coalescence using VSV

### 6.1 Introduction

The last few decades have seen simulation grow to become an essential tool for the study of contemporary materials and for the postulation of new materials [1–4]. The study of atomic clusters of small dimensions is gaining prominence because of their scientific and industrial importance [5, 6]. Molecular dynamics (MD) can be used to investigate many macroscopic behaviors such as crystalline cluster formation, defect formation and propagation, surface energies, molecular adsorption to surfaces, diffusion and other thermal effects [7, 8]. It is a useful method when the complexity or cost makes empirical study of the system difficult [4, 9]. Standard MD remains popular due to the continuing advances in computing and the ease of modelling the underlying force-fields [10–12]. The surface of a crystal is of considerable importance in materials because it interfaces mechanically and chemically with environments external to the crystal. The discontinuity of ordering at the surface makes it an active defect [13, 14]. Many composite material systems have precipitates in non-equilibrium compositions that lead to two important questions about the long term thermodynamic

stability of the non-equilibrium structure. It then becomes necessary to evaluate the diffusion of the various components in terms of atomic jump activation energies  $E_a$  and the Gibbs free energy  $\Delta E_g$  [15]. In general, for such systems three valuations can be done relative to the composition i.e. whether

1.  $E_a$  is low enough while  $\Delta E_g$  decreases monotonically,
2.  $E_a$  is low enough while  $\Delta E_g$  does not decrease monotonically, and
3.  $\Delta E_g$  is too high for atomic jumps.

The last case implies that diffusion is effectively not possible and that the non-equilibrium state is stable. The relative behaviour of  $E_a$  and  $\Delta E_g$  therefore plays an important role in diffusion, particularly for multi-component alloys.

In this study, the impact-oscillation responses of nano-sized Cu clusters were investigated using the developed VSV software [16, 17]. Specifically, the interactions between a 357-atom cluster and a separate sub-cluster of atoms having various initial positions and velocities were simulated at temperatures near 0K. The system is thus homogeneous and the considerations of the Gibbs free energy,  $\Delta E_g$  fall away. The initial simulations establish the cut-off distance by evaluating the minimum distance at which the cluster exerts an attractive force on an adatom, causing it to drift towards the cluster. This experiment is important because it establishes how a randomized cluster could spontaneously assimilate into a fcc crystal structure, thereby simulating spontaneous crystal growth. Structure degradation by localized forces that propagate into the lattice are used to simulate melting in response to adatoms projected into the main cluster. The parameters of the resulting cluster are compared with the known macro values for Cu. We also calculate the diffusion and other energetic parameters for the small cluster. The approximate dimensions of the cluster are  $7 \times 7 \times 7$  atoms, spanning a volume of approximately  $64 \text{ nm}^3$ . In the case where the adatom was projected into the lattice, the adatom was projected in the [100] direction. Presented below are the results of the

statistical analyses on the outputs of several repeated simulations. While the simulations are done with the main cluster near 0K, the average temperature of the main cluster is observed to rise when higher adatom projection velocities are used. There is a need to equilibrate the temperature of the lattice by randomizing particle velocities for a lattice that is assumed to be initially at 0K. We employ a relatively long integration time,  $\delta t$ , of 91.75 fs in order to calculate averaged static and low-frequency thermodynamic parameters of the cluster. Shorter integration times would permit the study of the higher frequency behaviour, such as phonon propagation and resonance. This study could lead to a better understanding of the mechanisms of metallic thin-film layer formation and the role of defects on surface thermodynamics.

## 6.2 The simulation model

The Finnis and Sinclair (FS) [18] potential was described in Chapter 3. Sutton-Chen (SC) [1] introduced an embedding energy  $E(\bar{\rho}_i)$  term into the FS potential to rather describe the energetic contribution of the local charge density. In the simulations, VSV code employs standard MD to evolve the position and momentum of each particle by making use of Verlet-Stormer Velocity integration at each time step [19]. The potential and simulation parameters are shown in Table 6.1.

Table 6.1 SC and simulation parameters.

| Parameter                    |                       |
|------------------------------|-----------------------|
| $\epsilon$                   | 0.012382 eV           |
| $m, n, c$                    | 6, 9, 39.432          |
| Integration time, $\delta t$ | 0.09175 ps (91.75 fs) |
| $t_{start}, t_{end}$         | 0, 91.75 ps           |
| $T$                          | 0 K                   |

## 6.3 Calculations

The simulations are in two broad groups, i.e.

- i. self-diffusion of single adatoms or multiple adatoms into the main cluster. This is done to investigate the force field that extends beyond the physical boundaries of the cluster by its effects on external adatoms
- ii. projection of adatoms into the main cluster to simulate adatom(s) that are at higher temperatures than the main lattice.

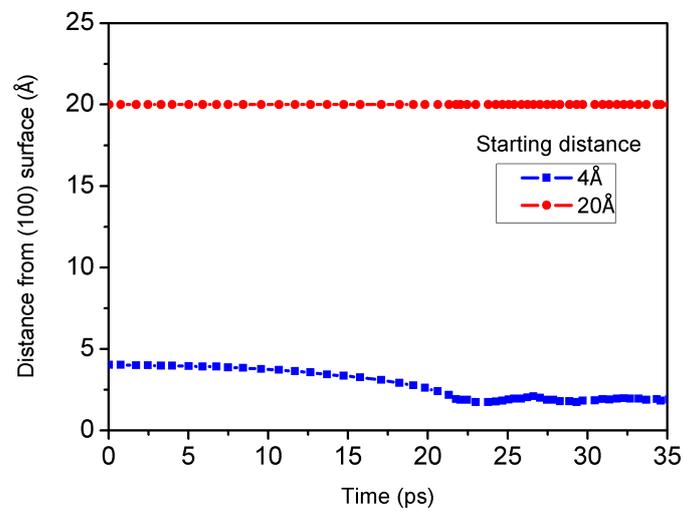
We assume ambient vacuum conditions around the interacting atoms, with each position and velocity recalculated in  $\delta t$  intervals.

### 6.3.1 Determination of cluster-adatom interaction distance

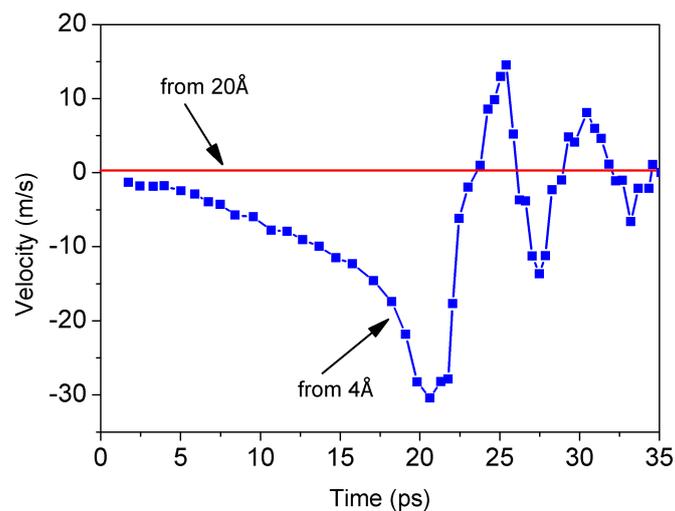
A cut-off radius,  $r_{cut}$ , is often used to speed up many body simulations with some trade-off in output accuracy. The first simulation below establishes the farthest distance at which the adatom begins to drift measurably towards the main cluster over the simulation time as an indication of  $r_{cut}(\text{min})$ . We have used this distance to estimate the diffusive migration activation energy i.e. the minimum energy required to effect the motion of the adatom. This energy exists because of a force field that extends some distance from the main cluster.

### 6.3.2 Single adatom diffusion

Figure 6.1 shows the simulated behavior of the adatom at 0K at two initial distances relative to the main cluster's (100) surface. In Figure 6.1a, the adatom from  $4\text{\AA}$  drifts accelerates towards the (100) and then settles at approximately  $1.8\text{\AA}$  of the surface, or  $2.55\text{\AA}$  of its nearest surface neighbors. Figure 6.2 shows the final position of the captured adatom. The arrow shows the initial direction of projection of the adatom. The final position of the adatom



(a) relative position



(b) actual velocity

Fig. 6.1 The graphs show the simulated time behaviors of an adatom that is initially at rest at two separate positions, 4 Å and 20 Å. In (a), the distance measured relative to the capturing (100) face and in (b), the adatom velocity.

is determined by the net effect of all the atoms and their velocities. Figure 6.3 shows the relative position on the cluster surface. Its position is stable up to the end of the simulation. This suggests that at that site it has minimum potential energy and is effectively captured. The adatom initially at 20 Å does not discernibly change its position relative to the main cluster

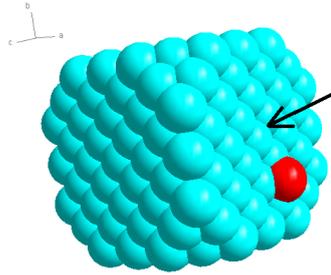


Fig. 6.2 Final position of captured adatom (red) from  $20\text{\AA}$ . The arrow shows the projection direction.

over the simulation time. Figure 6.1b shows the behavior of the adatom velocity in time, which supports the acceleration of from  $4\text{\AA}$  for a time of 22 ps. Close to the main cluster, both position and velocity of the adatom oscillate as it settles into its new captured position. The negative portions of the velocities show a reversal of the direction of the adatom at those time instants. The simulation was repeated for slightly displaced initial positions of the adatoms i.e. near both 4 and  $20\text{\AA}$ . Similar results were obtained where the captured position

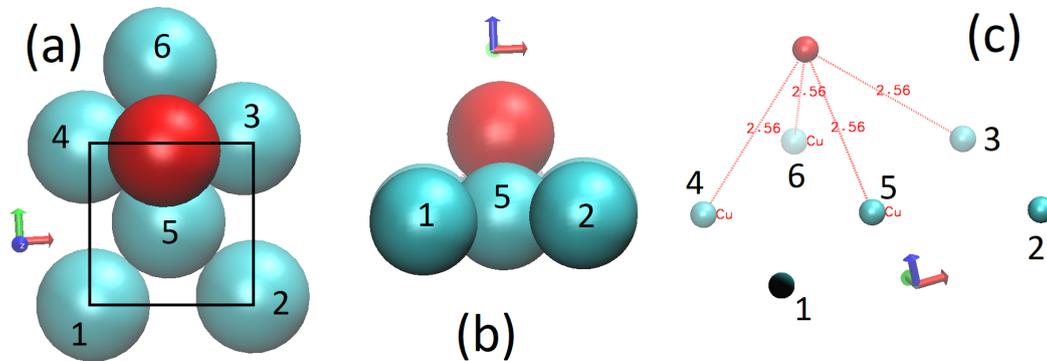


Fig. 6.3 Alternate views showing localized environment of the captured adatom (red) that was initially at  $4\text{\AA}$  above the surface. The reference square in (a) shows the reference positions of the (100) face atoms, (b) is the side view, (c) is the measurement view. The numbers show the same main lattice atoms in the different views.

is as depicted in Figure 6.3. This simulation suggests that for small clusters the application of  $r_{cut}(\text{min})$  should be avoided, or at least considered only beyond the effective force field

distance, such as  $20\text{\AA}$  in this case. For all subsequent calculations, a cutoff distance was not specified in the light of the small cluster size.

### 6.3.3 Multiple adatom diffusion

In the second set of simulations 10 Cu atoms at 0K, also shown in red in Figure 6.4a, are arbitrarily positioned at rest within  $4\text{\AA}$  of the (100) face of the main cluster. The new adatom positions are stable on the surface of the main cluster within 36.7 ps of the start of the simulation, as shown in Figure 6.4b. Figure 6.5 shows the absolute distance behaviour of

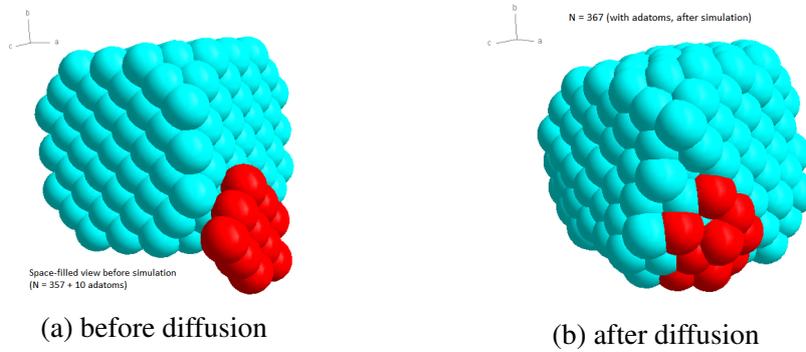


Fig. 6.4 Space-filled representation of the diffusive capture of ten adatoms on a (100) face.

each of the 10 adatoms as a function of time. This distance is calculated relative to the origin for adatom  $j=\{1,2,\dots, 10\}$  according to

$$|\bar{r}_j| = (x_j^2 + y_j^2 + z_j^2)^{1/2}. \quad (6.1)$$

Figure 6.5 shows that the adatoms diffuse follow a number of step-plateau changes over the simulation time. For instance, for all adatoms, in the simulation shown, the initial positions change rapidly at 9.2 ps, 55 ps and 90 ps. Figure 6.4b shows the resulting cluster corresponding to Figure 6.5. The lattice exhibits coalescence of Cu atoms onto the surface of the main cluster, with an average simulated separation of  $2.55\text{\AA}$  between any atom and its neighbor in the resulting structure. This distance is also the near neighbor distance

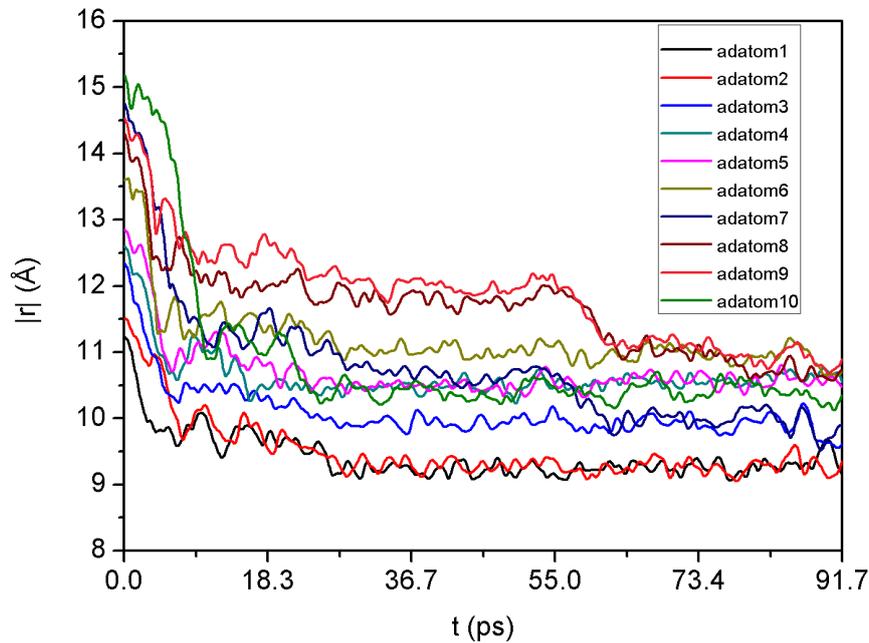


Fig. 6.5 Simulated distances of adatoms from an origin at the center of the cluster.

in purely crystalline fcc Cu. Structure formation by coalescence of metallic particles has also been observed in many simulation studies of small dimensioned clusters in other simulation approaches, such as multi-resolution MD (MRMD) and Kinetic Monte Carlo (KMC) methods [20, 21]. Using MRMD, Saitoh et al [22] found agreement with ordinary MD results with respect to the resultant shapes and trajectories and found a lower increase in net cluster kinetic energy after coalescence. MRMD accommodates rigid body dynamics in an otherwise ordinary MD simulation using the idea that atomic or molecular particle motion are effectively surface decided phenomena [4, 23]. KMC has been used to simulate surface diffusion and thin film growth and is thought to be useful because it allows experimentally congruent time scales to be simulated. However, it is limited in its predictive power because it needs a priori specification of atomic stimuli and energy constraints. The work of studied Doye and Wales [24] can be considered a demonstration of cluster coalescence using the minimum potential energy surfaces. The results in Figure 6.5 suggest that the coalescence is

a step-wise process as Figure 6.6 depicts using approximate boundaries of the first and tenth adatom radial distances shown in 6.5. Takahashi et al [25, 26] also report this apparent step-wise coalescence in clusters of silver atoms during their simulation search for consistency between atomistic and continuum models. They use the Rosato, Guillope and Legrand (RGL) many body potential for Ag, which has few fitting parameters than standard EAM models [27, 28]. Gafner et al [29] have also studied structure formation in Au, Cu and Ni nanoclusters ranging in size from 16 to 50 Å using a tight-binding (TB-SMA) potential. They reported that structure formation is influenced by the particle size. Rapid coalescence is thought to be due to boundary formation followed by rapid stabilization, over three critical stages:

- i. incubation, where the interaction forces between two clusters are communicated on the whole
- ii. rigid motion approach, the attractive forces overcome random thermal motions in the cluster, and
- iii. boundary formation itself, which requires enough thermal energy to displace atoms.

Similar, qualitative results that differ only over the timescales have also been reported for gold and nickel [30, 31].

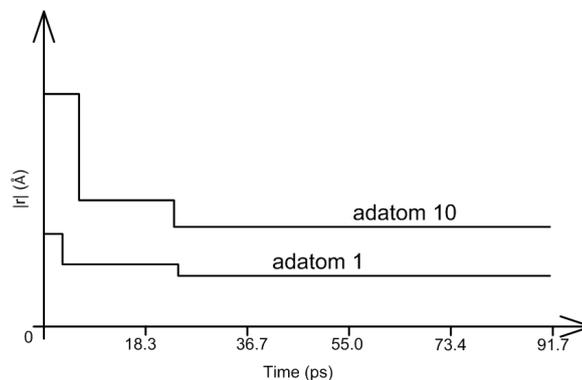


Fig. 6.6 Depiction of stepwise coalescence of diffusing adatoms.

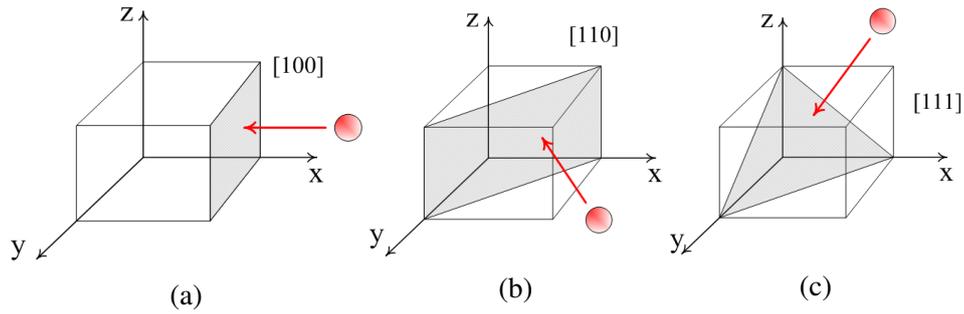


Fig. 6.7 Diagrams showing the direction of projection of the adatom, towards the shaded plane within the main cluster, which is outlined by the box.

### 6.3.4 Projected adatoms

The effects of projecting adatoms into the main cluster at different velocities were also simulated using an energy equi-partition theorem (EEPT) thermostat. The main lattice was initialized to 0K as before. With multiple projected adatoms, each projection velocity was randomized using a Gaussian function such that the total kinetic energy was fixed for each simulation. The velocities follow the Maxwell-Boltzmann distribution (MBD) [32] provided that an exponential randomization factor  $f$  be found such that

$$f = (k_B T / m)^{1/2},$$

where  $k_B$  is Boltzmann constant [33–35]. For simulations done at 0K,  $e^f = 1$ . In EEPT, temperature and expectation velocity  $v_j$  are equivalent, as given by Eq. (7.3). Using this model, we have calculated the effects of projecting single/multiple adatoms into the main cluster, depicted in Figure 6.7, under conditions of purely translational, irrotational atom motion, i.e. with  $N_f = N$ .

#### 6.3.4.1 Projected single adatoms

Figure 6.8 shows the position-time behavior of the adatom. The calculations show that an adatom speed above approximately 108 m/s, causes the adatom to be deflected beyond the

cluster. The equivalent temperature using Equation (7.3) is approximately 88.7K. Hence the

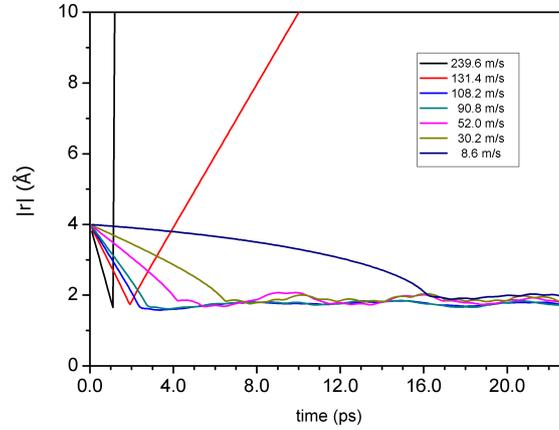


Fig. 6.8 Calculated absolute distance-time behavior of single adatom of various velocities.

capture of Cu atoms occurs at speeds lower than 108 m/s i.e. initial energies below 3.08 eV. This result corresponds to the per-atom cohesive energy,  $E_{coh}$ , of copper. Cohesive energy is the amount of energy required to add an atom from a point at infinity to a crystal at 0K [11, 36]. Figure 6.9 shows the calculated potential variation relative to the (100) surface

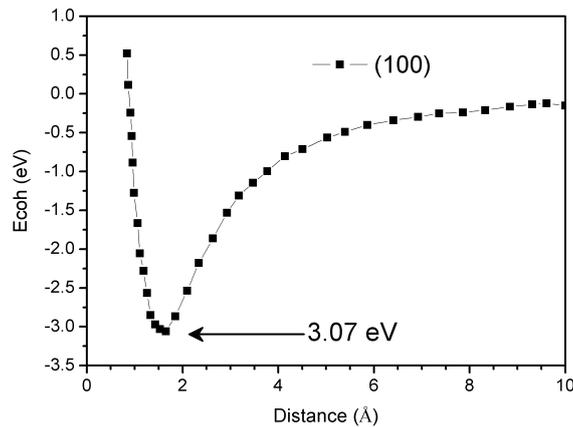
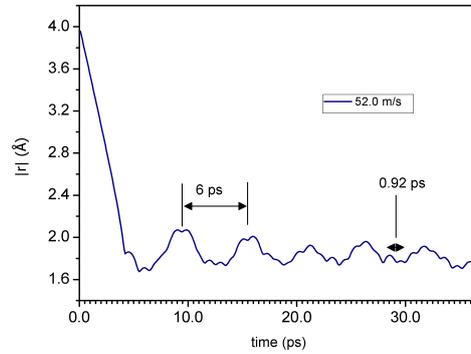


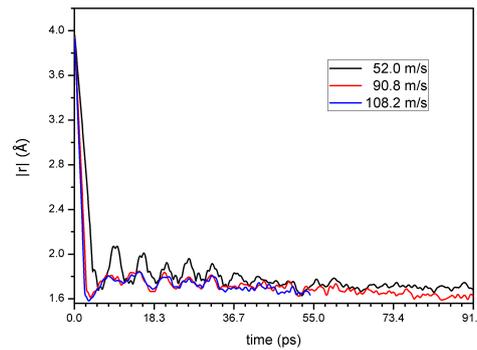
Fig. 6.9 Potential distribution calculated for the (100) surface in an extensive Cu crystal.

of an extensive Cu crystal. The localities of the potential well in Figure 6.9 indicate the most likely capture sites for the adatom. Such a site has been established in a foregoing

section and is depicted in Figure 6.3. Figure 6.10a shows strong oscillations in the position of the adatom at 52 m/s. We suggest this as an alternative method to calculate the vacancy



(a)  $v = 52 \text{ m/s}$



(b)  $v=52.0, 90.8, 108.2 \text{ m/s}$

Fig. 6.10 Adatom-near neighbor bond length oscillations during adatom capture on surface.

formation energy of the crystal through a deterministic dynamic approach. Adatoms having energies below a threshold value are captured into a stable position and cannot penetrate deep into the crystal bulk. Adatoms having higher energies by virtue of their speeds (i.e. thermal energies) may either penetrate deeper into the crystal or are deflected beyond the confines of the crystal. The amplitudes of the oscillations are lower and decay faster in comparison with those at 52 m/s, as shown in Figure 6.10b. The approximate period of oscillation is 6 ps, but vibrations depict higher frequency component of 0.92 ps period. This can be explained as faster vibrations along the projection direction, [100] for the adatom impacts the crystal

more or less on a (100) face. Figure 6.11 shows the component resolution of adatom to near neighbor bond length oscillations for an adatom projection speed of 52 m/s. The oscillations begin well before the adatom is assimilated onto the surface of the cluster. This is expected from the interplay of the attractive and repulsive forces from the main cluster.

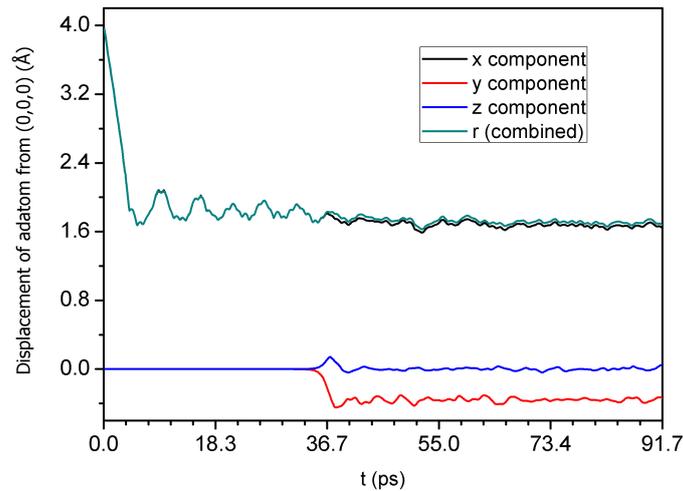


Fig. 6.11 Resolved displacement ( $x$ ,  $y$ ,  $z$ ) components of radial displacement for 52 m/s adatom.

This shows that the energy transfer is expected along the  $x$  direction in contrast to the  $y$  and  $z$  directions. These oscillations indicate the presence of lattice elasticity. This implies that the lattice promotes mechanical waves or phonons [37, 38].

#### 6.3.4.2 Projected multi-adatoms

In the third case, 16 Cu atoms at 88.7K are positioned within  $4\text{\AA}$  of the (100) face of the main cluster which is at 0K. Figure 6.12 shows the configuration, (a) before, and (b) after simulation. Only 3 adatoms are in stable surface positions on the main cluster while 13 adatoms were scattered beyond. The simulations show that the captured adatoms populate the surface and none have penetrated the bulk. The average distance between all three adatoms with their five nearest neighbors is  $2.48\text{\AA}$ , in contrast to the case of the single adatom above

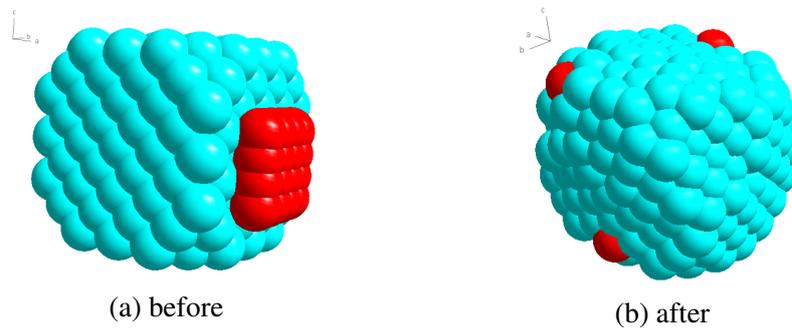


Fig. 6.12 Depiction of initial and simulated atom positions after projection towards a (100) face.

and the expected 0K Cu near-neighbor distance of  $2.55\text{\AA}$ . The new lattice has 360 atoms in total, suggesting a by-mass crystal growth of 0.83%. This simulation suggests a method to calculate the scattering effect of a lattice of atoms having different energies. This could be used to simulate sputtering processes on the surfaces of fcc lattices. Figure 6.12 also depicts a more pronounced coalescence of the main cluster. Using Equation (7.3) with  $N_f=360$ , the calculated overall cluster temperature was 0.22 K.

When capture occurs the final average speed of the adatom approaches zero, with the adatom attaining its potential energy minimum. Kinetic energy is transferred to the expand or contract lattice bonds readjustments, hence increasing the elastic potential energy of the cluster. This causes sustained oscillations and a generalized temperature rise. Lattice vibrations propagate both as bulk and surface phonons. Since the calculations assume an isolated harmonic oscillator with in this case 360 purely translational degrees of freedom, these oscillations are expected indefinitely since there are no energy loss mechanisms in the simulation.

### 6.3.5 Simulating lattice assimilation growth

Repeating the above procedure of atom capture using more adatoms effectively simulates lattice growth. This demonstrates a further advantage of the VSV toolkit over existing third party software.

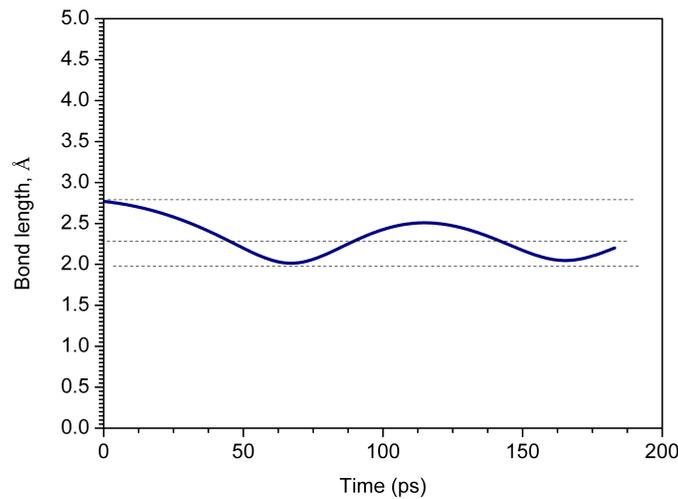


Fig. 6.13 Distance between adatom and its nearest neighbor over time showing oscillations at the cluster surface.

The plots in Figure 6.14 show the growth of a lattice at 20 ps intervals. The additional adatoms are initially at rest and placed close to the main cluster and the force field redistributes the atoms gradually until they are in the final positions shown. Only three atoms are successfully assimilated in this configuration.

The final relative positions of all atoms change for the same reason of reassertion of minimum potential energy. For this reason, the final main lattice appears in all cases to be considerably deformed in comparison with its starting shape shown in Figure 6.14a. The deformation of the lattice appears to suggest that the presence of isomorphic forces per unit length whose effects on the simulation become more manifest as simulation time progresses. An analogy is the formation of curvature in a droplet of water due to surface tension. For a much wider and larger main cluster, this less deformation may be expected to be less

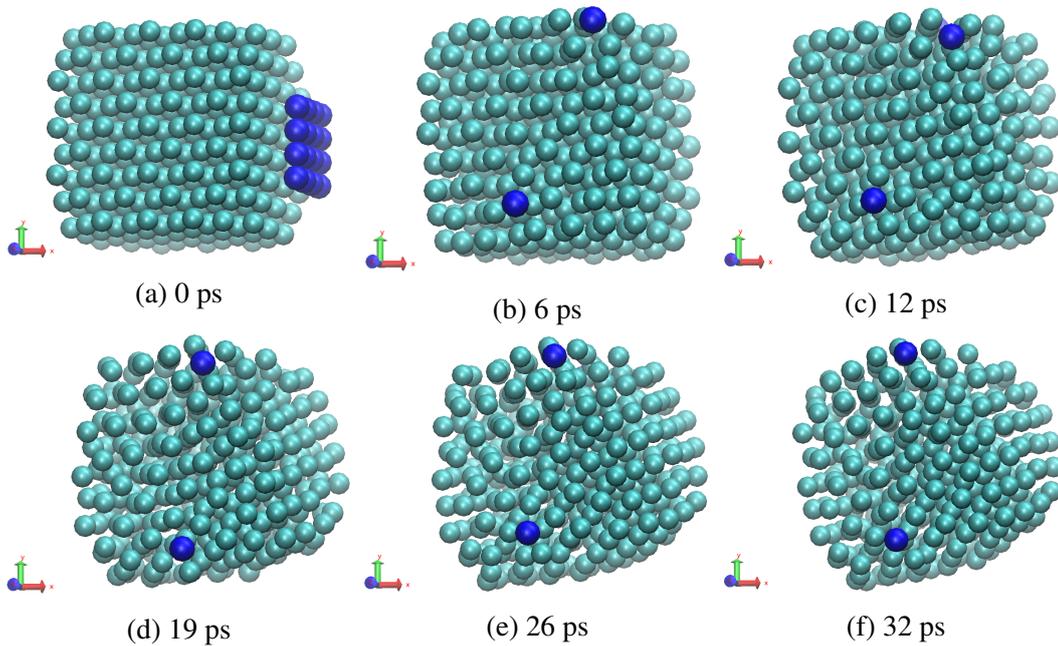


Fig. 6.14 Simulation of crystal growth by assimilation of adatoms.

pronounced farther away from the adatom capture sites. The initial shape of the main cluster is not expected to be preserved due to the equalization of the potential energies. The average separation between the adatoms with any of their neighbors, which can be other adatoms or main lattice atoms was calculated at  $2.55\text{\AA}$ . This coincides with the distance between corner and face-centered atoms in the Cu fcc unit cell. Figure 6.15 shows the near-neighbour distances of the captured atoms. The global minima approach of Doye and Wales [24] for 3 to 80 atoms at given  $m$ - $n$  factors shows that though inherently crystalline, the most stable arrangement may not immediately be recognizable visually as being an fcc structure despite being one. Thus far, our preliminary calculations using VSV [16] have reproduced their results. The present results are a reasonable extension of their results to 357 and subsequently 2281 atoms.

### 6.3.6 Surface diffusion

Self-diffusion of fcc metals, notably Cu, Ni,  $\alpha$ -Fe has been studied extensively using different empirical methods [39–41]. Most such studies are done at increased temperatures that are motivated by two factors. Firstly, the effects of atom diffusion that may lead to system failures need quantification in conditions where high temperatures are expected. Secondly, in measurements, the observability of diffusion vastly improves with temperature. Many of these studies rely on tracer atoms doped into the lattice of interest whereupon the diffusion parameters are deduced indirectly through tracer diffusion [42–44]. Although computational studies of diffusion have been done, they have tended to mirror empirical studies in their boundary conditions but for systems with particle numbers on the macroscale. Studies of diffusion temperatures and metallic nanoclusters are relatively few. Hence there are few references pointing towards low temperature diffusion. If one uses the reasoning that the pairwise forces and embedding potentials do not cease even at 0K, then particles should possess a potential to reorganize, with energies that are comparable to the Fermi level of the

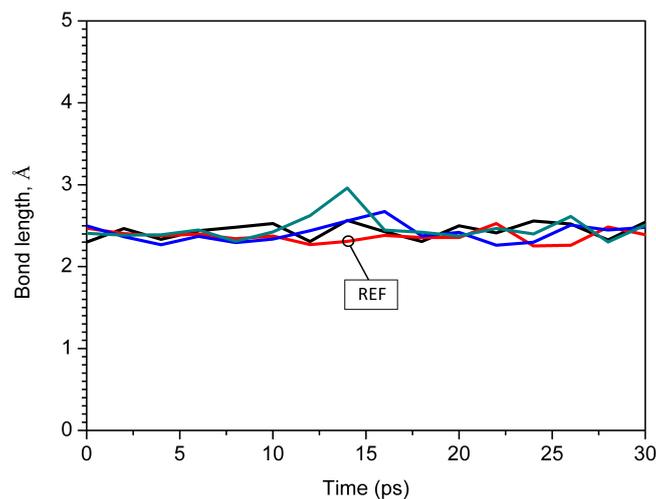


Fig. 6.15 Near-neighbor distance between the three captured copper atoms. The fourth plot, labeled “REF”, is a reference plot based on the bond length between two randomly selected adjacent Cu atoms that were initially in the main cluster.

atom. In short, the potential to diffuse should continue to exist at 0K. In the absence of such studies, the questions that naturally arise are then whether the underlying mechanisms of particle movement and the parameters such as diffusivity and activation energy as calculated using the classical approaches lead to values that are correct and comparable to those of the bulk material, or whether a completely new model is needed.

Diffusion is the main mass transfer mechanism in solids that occurs down a concentration gradient and depends on temperature. It involves step-wise atom migration and can be calculated using Eq. (3.28). In the general treatment, diffusion is understood to occur either through self-diffusion within the bulk or through surface diffusion. Self-diffusion is highly temperature dependent because the concentration of the vacancies themselves depend on temperature. Diffusion flux,  $J$ , is the number of diffusing particles that cross a unit area that is perpendicular to the direction of particle motion in unit time, i.e.

$$J = D \frac{\partial c}{\partial x}, \quad (6.2)$$

where  $D$  is a temperature dependent diffusion coefficient that can be written in the Arrhenius form i.e.

$$D = D_0 e^{-Q/RT}, \quad (6.3)$$

where  $Q$  is the molar diffusion activation energy and  $R$  is ideal gas constant [12, 42].

Figure 6.16 plots the instantaneous value of  $\sigma^2$  in Equation (3.28) for the atom configuration in Figure 6.4b. The figure shows two regimes involved in  $\sigma^2$  that are consistent with the simulated behavior in Figures 6.5 and 6.6. In the first region, labeled A, the adatoms start from rest and accelerate towards the main cluster and begin to integrate into it within first 36 ps. The calculated diffusivity is seen to rise rapidly at first, peaks and then declines rapidly in this time. As the cluster atoms reorder, the instantaneous local temperatures calculated according to Eq. (7.3) will change. Diffusivity is known to increase with temperature. The

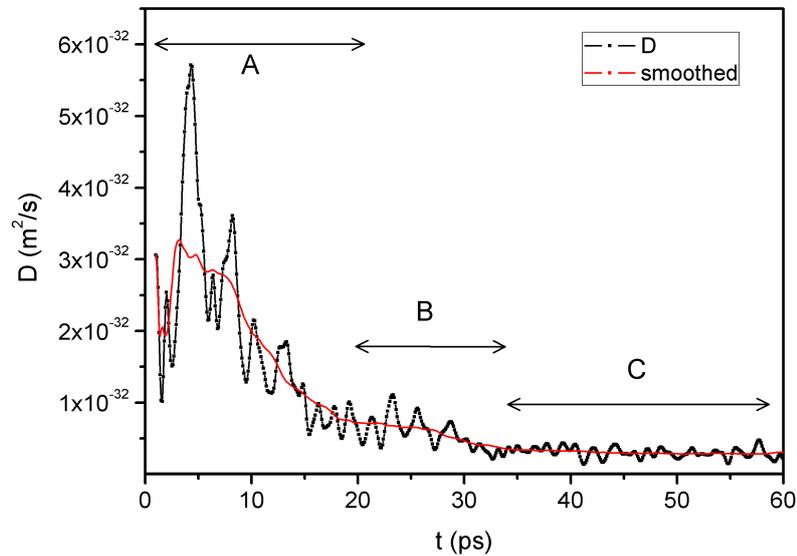


Fig. 6.16 Calculated diffusivity in Cu using 10 self-diffusing adatoms.

rapidly changing value of  $D$  in region A suggests that the temperature also rapidly increases momentarily due to the approaching adatoms. Subsequent simulations show a temperature behaviour that appears to match the trend in diffusivity in Figure 6.16. We have taken the maximum of  $D$  as  $D_0$  in Eq. (6.3). The averaged diffusivity plot of  $D$  versus time exhibits three distinct regions, A, B and C although the overall trend appears to be an exponential decay. Region C is replotted in Figure 6.17 and clearly shows the exponential trend.

In region B, the integration of the captured adatoms into the main cluster begins and evolves towards coalescence as the new cluster attempts to reassert a new potential energy minimum. The atom motions in B are vibratory about their relative lattice positions due to the natural frequencies of the lattice due to the net forces due to all the neighbouring atoms. The underlying mechanism of coalescence is diffusion by Brownian motion. In the coalesced cluster, it may be expected that the concentration of atoms equalizes to limit further diffusion. The increased disorder in coalescence is equivalent to melting at much lower temperatures [45], leading to the generally accepted conclusion that the melting points of nanoparticles

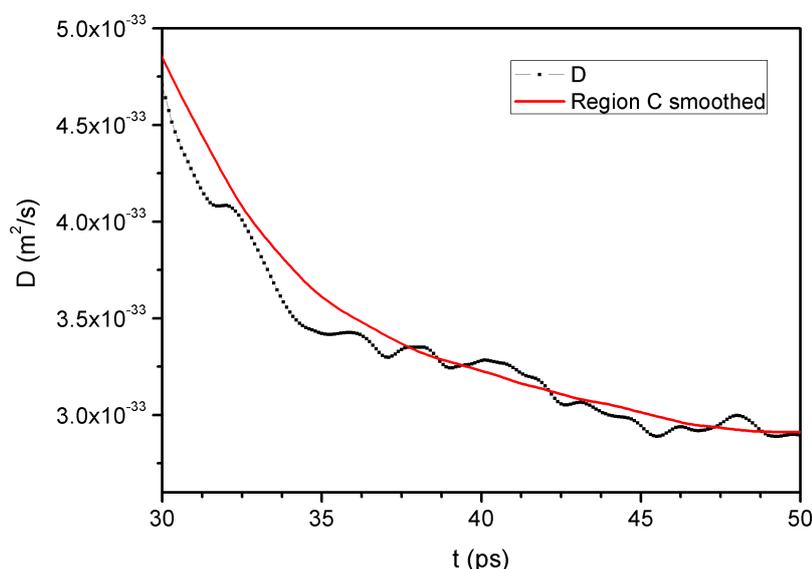


Fig. 6.17 Graph showing an approximate exponential decay of calculated diffusivity over time.

are much lower than for the bulk. The extent of the temperature difference depends on the particle size. This behavior has been reported from experiments on Au nanoparticles [46], and theoretically for surface phenomena using an atomistic model [47].

We take the approach of calculated the diffusion flux  $J$  by first defining a cross-sectional area by using an arbitrary plane located in the cluster, such as [110] and counting the number of atoms that cross the plane in 10 ps. This was repeated for three planes. This calculation did not involve a new simulation but post-processing of simulation output.

To estimate  $D_0$  and  $Q$  using the classical approach, we have assumed that the instantaneous temperature also decays exponentially and asymptotically with time towards a steady overall cluster temperature of 0.22 K as calculated using EEPT. Calculations on the data of Figures 6.16 and 6.17, give  $5.5 \times 10^{-32} \text{ m}^2/\text{s}$  and 291 kJ/mol for  $D_0$  and  $Q$  respectively for self diffusivity and activation energy. The available references are listed at high temperatures. The *Smithells Metals Reference Book* [48] gives Cu self-diffusion parameters at 500 °C as

211 kJ/mol (2.19 eV/atom) for activation energy, and  $7.8 \times 10^{-5}$  and  $4.2 \times 10^{-5}$  m<sup>2</sup>/s for  $D_0$  and  $D$ . Our calculated  $Q$  determined corresponds to 3.01 eV/atom.

## 6.4 Conclusions

We have carried out standard MD simulations on a small cluster of up to 373 copper atoms using the Sutton-Chen embedded atom potential in a vacuum environment. The study highlights some crystal formation dynamics and surface growth mechanisms through the capture of additive atoms. The size of the fcc cluster is kept small to better assess the effects of the interactions. The interactions studied in this present article relate to self-diffusion and low energy projection of atoms towards the surface of a larger cluster. The main cluster was confined to 0 K. The simulations provide evidence of a spontaneous diffusive formation of surface structure and spontaneous coalescence at near 0K temperatures, suggesting that diffusion at low temperatures is driven by pairwise force interactions rather than by Brownian motion. The calculations also show the adatom maintains an equilibrium distance with its new near neighbors that oscillates with a low amplitude around the 2.6 Å. In addition, the calculations also suggest that coalescence of the main cluster occurs under certain conditions as the potential energy minimum is reasserted. In the case of the initial arrangements at rest, the final temperature is observed to rise to a steady temperature of approximately 0.22 K, which is beyond the margin of error. The diffusion activation energy is calculated at 3.01 eV/atom against the 2.19 eV/atom literature value. The diffusivity is typically ascertained empirically at high temperature and there are no literature values for cryogenic temperatures. However, using the standard diffusion approach it was estimated at  $5.5 \times 10^{-32}$  m<sup>2</sup>/s at 0.22 K.

# References

- [1] Sutton AP, Chen J. Long-range Finnis-Sinclair potentials. *Philosophical Magazine*, 63 (1):139–156, 1990.
- [2] Car R, Parrinello M. Unified approach for molecular dynamics and density functional theory. *Phys. Rev. Lett.*, 55(22):2471–2474., 1985.
- [3] Car R, Parrinello M. The unified approach for molecular dynamics and density functional theory, in simple molecular systems at very high density. in *NATO ASI Series, Series B, Physics*, P.P. Loubeyre and N. Boccara (Eds.), 186:455–476, 1989.
- [4] Abraham MJ, Murtola T, Schulz R, Pall S, Smith JC, Hess B, Lindahl E. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 101:88–95, 2015. doi: 10.1016/j.softx.2015.06.001.
- [5] Remler DK, Madden PA. Molecular dynamics without effective potentials via the Car-Parrinello approach. *Mol. Phys.*, 70(6):921–66, 1990.
- [6] Tuckerman ME. Ab initio molecular dynamics: basic concepts, current trends and novel applications. *J. Phys.: Condens. Matter.*, 2002. URL [stacks.iop.org/JPhysCM/14/R1297](http://stacks.iop.org/JPhysCM/14/R1297).
- [7] Sebastian IS, Aldazabal J, Capdevila C, Garcia-Mateo C. Diffusion simulation of CrFe bcc systems at atomic level using a random walk algorithm. *Phys. Stat. Sol. (a)*, 205(6): 1337–1342, 2008. doi: 10.1002/pssa.200778124.

- [8] Jian Min Z, Fei M, Ke-Wei X. Calculation of the surface energy of fcc metals with modified embedded-atom method. *Applied Surface Science*, 13(7):34–42, 2004. doi: 10.1016/j.apsusc.2003.09.050.
- [9] Mendeleev MI, Han S, Srolovitz DJ, Ackland GJ, Sun DY, Asta M. Development of new interatomic potentials appropriate for crystalline and liquid iron. *Philosophical Magazine*, 83(35):3977–3994, 2003. doi: 10.1080/14786430310001613264.
- [10] Das A, Ghosh MM. MD simulation-based study on the melting and thermal expansion behaviours of nanoparticles under heat load. *Computational Materials Science*, 101: 88–95, 2015. doi: 10.1016/j.commatsci.2015.01.008.
- [11] van der Walt C, Terblans JJ, Swart HC. Molecular dynamics study of the temperature dependence and surface orientation dependence of the calculated vacancy formation energies of Al, Ni, Cu, Pd, Ag, and Pt. *Computational Materials Science*, 83:70–77, 2014. doi: 10.1016/j.commatsci.2013.10.039.
- [12] Terblans JJ. Calculating the bulk vacancy formation energy (ev) for a Schottky defect in a perfect cu(111), cu(100) and a cu(110) single crystal. *Surf. Interface Anal.*, 33: 767–70, 2002. doi: 10.1002/sia.1451.
- [13] Vlieg E. The role of surface and interface structure in crystal growth. *Progress in Crystal Growth and Characterization of Materials*, 62(2):203–211, 2016. doi: 10.1016/j.pcrysgrow.2016.04.010.
- [14] Griebel M, Knapek S, Zumbusch G et al (Eds.). Numerical simulation in molecular dynamics. in, *Texts in Computational Science and Engineering 5*, Springer, Berlin, 5 (ISBN 978-3-540-68094-9), 2007.

- [15] Angelina Orthacker, Georg Haberfehlner, Johannes Tändl, Maria Cecilia Poletti, Bernhard Sonderegger, and Gerald Kothleitner. Diffusion defining atomic scale spinodal decomposition within nano-precipitates. *Nature materials*, 17, 2018.
- [16] Ocaya R, Terblans JJ. Temperature specification in atomistic molecular dynamics and its impact on simulation efficacy. *Paper presented at CCP2016 - 28th IUPAP Conference on Computational Physics; Johannesburg, South Africa; 2016 July 10-14, 2016.*
- [17] Ocaya R, Terblans JJ. Addressing the challenges of standalone multi-core simulations in molecular dynamics. in *P. Ramasami (ed.), Computational Sciences, De Gruyter, 2017.*
- [18] Sutton AP, Chen J. Long-range Finnis-Sinclair potentials. *Philosophical Magazine Letters*, 61(3):139–146, 1990. doi: 10.1080/09500839008206493.
- [19] Ocaya R, Terblans JJ. C-language package for standalone embedded atom method molecular dynamics simulations of fcc structures. *SoftwareX*, 5:107–111, 2016. doi: 10.1016/j.softx.2016.05.005.
- [20] B. D. Butler, Gary Ayton, Owen G. Jepps, and Denis J. Evans. Configurational temperature: Verification of monte carlo simulations. *The Journal of Chemical Physics*, 109(16):6519–6522, 1998. doi: 10.1063/1.477301.
- [21] Talat S. Rahman, Abdelkader Kara, Altaf Karim, and Oleg Trushin. Cluster diffusion and coalescence on metal surfaces: applications of a self-learning kinetic monte-carlo method. *MRS Proceedings*, 859:JJ8.4, 2004. doi: 10.1557/PROC-859-JJ8.4.
- [22] K.-I. Saitoh, M. Komaya, and T. Inaba. Multi-resolution molecular dynamics method for coalescence process of metallic atom clusters. *Nippon Kikai Gakkai Ronbunshu, A*

- Hen/Transactions of the Japan Society of Mechanical Engineers, Part A*, 68(2):210–216, 2002.
- [23] A.M. Mazzone. Coalescence of metallic clusters: A study by molecular dynamics. *Philosophical Magazine B: Physics of Condensed Matter; Statistical Mechanics, Electronic, Optical and Magnetic Properties*, 80(1):95–111, 2000. doi: 10.1080/13642810008218342.
- [24] Doye JPK, Wales DJ. Global minima for transition metal clusters described by the Sutton-Chen potentials. *New J. Chem.*, pages 733–744, 1998.
- [25] Carter W.C. Takahashi A.R., Thompson C.V., . URL <http://hdl.handle.net/1721.1/3968>. Atomistic Simulations of Metallic Cluster Coalescence.
- [26] Carter W.C. Takahashi A.R., Thompson C.V., . URL <http://hdl.handle.net/1721.1/3668>. Metallic Cluster Coalescence: Molecular Dynamics Simulations of Boundary Formation.
- [27] V. Rosato, M. Guillopé, and B. Legrand. Thermodynamical and structural properties of f.c.c. transition metals using a simple tight-binding model. *Philosophical Magazine A: Physics of Condensed Matter; Structure, Defects and Mechanical Properties*, 59(2): 321–336, 1989. doi: 10.1080/01418618908205062.
- [28] M. Guillopé and B. Legrand. (110) surface stability in noble metals. *Surface Science*, 215(3):577–595, 1989. doi: 10.1016/0039-6028(89)90277-X.
- [29] Y.Y. Gafner, S.L. Gafner, Z.V. Golonenko, L.V. Redel, and V.I. Khurstalev. Formation of structure in Au, Cu and Ni nanoclusters: MD simulations. In *IOP Conference Series: Materials Science and Engineering*, volume 110, pages 1–5, 2016. doi: 10.1088/1757-899X/110/1/012015.

- [30] L.J. Lewis, P. Jensen, and J. Barrat. Melting, freezing, and coalescence of gold nanoclusters. *Physical Review B - Condensed Matter and Materials Physics*, 56(4): 2248–2257, 1997. doi: 10.1103/PhysRevB.56.2248.
- [31] H. Zhu and R.S. Averback. Sintering processes of two nanoparticles: A study by molecular dynamics simulations. *Philosophical Magazine Letters*, 73(1):27–33, 1996. doi: 10.1080/095008396181073.
- [32] Ocaya R, Terblans JJ. Coding considerations for standalone molecular dynamics simulations of atomistic structures. *Paper presented at CCP2016 - 28th IUPAP Conference on Computational Physics; Johannesburg, South Africa; 2016 July 10-14, 2016.*
- [33] Knuth D. *The art of computer programming, seminumerical algorithms*. Addison-Wesley, 1997.
- [34] Box G and Muller M. A note on the operation of random normal deviates. *Ann. Math Stat.*, 29:610–611, 1958.
- [35] Marsaglia G. The role of surface and interface structure in crystal growth. *Proc. Nat. Acad. Sci.*, 61:25–28, 1968.
- [36] C. Kittel. *Introduction to Solid State Physics*. Wiley, 2004. ISBN 9780471415268. URL <https://books.google.co.za/books?id=kym4QgAACAAJ>.
- [37] Ditlevsen Peter D, Nømskov Jens K. Vibrational properties of aluminum, nickel and copper surfaces. *Surface Science*, 254:261–274, 1991. doi: 10.1.1.718.7356.
- [38] Nelson JS, Daw MS, Sowa EC. Cu(111) and Ag(111) surface-phonon spectrum: The importance of avoided crossings. *Phys. Rev. B*, 40:1465, 1989.

- [39] D.B. Butrymowicz, J.R. Manning, and Read M.E. Diffusion in copper and copper alloys. Part I. volume and surface self-diffusion in copper. *Journal of Physical and Chemical Reference Data*, 2(3):643–656, 1973. doi: 10.1063/1.3253129.
- [40] N.L. Peterson. Self-diffusion in pure metals. *Journal of Nuclear Materials*, 69-70:3 – 37, 1978. ISSN 0022-3115. doi: 10.1016/0022-3115(78)90234-9.
- [41] A. Kuper, H. Letaw, L. Slifkin, E. Sonder, and C. T. Tomizuka. Self-diffusion in copper. *Phys. Rev.*, 96:1224–1225, Dec 1954. doi: 10.1103/PhysRev.96.1224.
- [42] Chelsey Z. Hargather, Shun-Li Shang, Zi-Kui Liu, and Y. Du. A first-principles study of self-diffusion coefficients of FCC Ni. *Computational Materials Science*, 86:17 – 23, 2014. ISSN 0927-0256. doi: 10.1016/j.commatsci.2014.01.003.
- [43] Marek Zajusz, Juliusz Dąbrowa, and Marek Danielewski. Determination of the intrinsic diffusivities from the diffusion couple experiment in multicomponent systems. *Scripta Materialia*, 138:48 – 51, 2017. ISSN 1359-6462. doi: 10.1016/j.scriptamat.2017.05.031.
- [44] Chamati H, Papanicolaou NI, Mishin Y, Papaconstantopoulos DA. Embedded-atom potential for Fe and its application to self-diffusion on Fe(100). *Surface Science*, 600: 1793–1803, 2006. doi: 10.1016/j.susc.2006.02.010.
- [45] M. José-Yacamán, C. Gutierrez-Wing, M. Miki, D.-Q. Yang, K. N. Piyakis, and E. Sacher. Surface diffusion and coalescence of mobile metal nanoparticles. *The Journal of Physical Chemistry B*, 109(19):9703–9711, 2005. doi: 10.1021/jp0509459.
- [46] Ph. Buffat and J-P. Borel. Size effect on the melting temperature of gold particles. *Phys. Rev. A*, 13:2287–2298, 1976. doi: 10.1103/PhysRevA.13.2287.

- 
- [47] C.Q. Sun, Y. Wang, B.K. Tay, S. Li, H. Huang, and Y.B. Zhang. Correlation between the melting point of a nanosolid and the cohesive energy of a surface atom. *Journal of Physical Chemistry B*, 106(41):10701–10705, 2002. doi: 10.1021/jp025868l.
- [48] E.A. Brandes and G.B. Brook, editors. *Smithells Metals Reference Book*. Elsevier, 7th ed. edition, 1992. ISBN 9780080517308. doi: 10.1016/C2009-0-25363-3.

# Chapter 7

## Detection of lattice phonons and their propagation in VSV

### 7.1 Introduction

In this chapter we suggest and apply VSV to simulate the phonon propagation in a novel way. We show that this approach could lead to the clarification of the dynamics and thermal conductivity nanosized homogeneous atom clusters. We devise a novel method to simulate arbitrary phonons and follow their propagation on the surfaces and the bulk under the formalism of the Sutton-Chen embedded atom model. We show that the coupled oscillations result in both surface and bulk longitudinal, transverse and shear waves in the lattice. We calculate the transient behavior of various bond lengths with time and apply fast Fourier transformations to highlight the frequency behavior of the wave. Finally, we show that the frequency spectral, elastic constants and thermal effects are correctly reproduced as compared with experimental data.

Dispersion of surface and bulk phonons in different crystals has been studied extensively using both empirical and computational approaches with varied degrees of success. The theory of phonon propagation in crystals due to elasticity is therefore well developed.

Computational methods have ranged from force methods of varied complexities to *ab initio* methods [1]. These studies have shown that the surface and bulk dispersion properties can differ substantially [2, 3]. In a recent review by Rassoulinejad-Mousavi et al [4], force field suitability assessments on current interatomic potentials were done for different Cu, Ni and Al at room temperature using extensive databases. These assessments were focused on the calculation of elastic moduli and suggested that users of existing models pay attention to the accuracy of the elastic constants.

Three main methods are generally used, namely the slab technique (ST), Green function (GF) methods, and MD methods. In ST, surface vibrations are evaluated through a direct calculation of the dispersion modes and frequencies [5, 6]. ST has been used widely in the microscopic domain. Here, the macromaterial is defined as a slab of constrained surfaces consisting of a finite number of atoms whose eigenvector displacements are calculated. In the GF method, the effect of massive forces which are beyond the limits of a modulus of elasticity are calculated. In the MD method, macroscopic behaviors are calculated as aggregates of atomic contributions. Certain phenomena, such as low-amplitude driven phase transitions, surface melting and disordering can be studied through MD simulation are relatively easier to calculate using MD [7, 8]. However, a major limitation of MD has been its inherent dependence on computational power [9], making ST and GF methods usually the first consideration in a given dispersion problem. An advantage of MD is that the vibrational modes can be evaluated directly from the temperature distribution. However, the risk of simultaneous sampling of different points in the reciprocal lattice is an issue in MD. This problem can be lessened by using different sized surfaces to account for the wavevector dependence of the dispersion relations. Another inherent weakness of MD is that it has thus far not readily yielded atomic displacements from calculated spectral densities [1].

This chapter has two aims. First, we apply a previously developed toolkit based on the Sutton-Chen (SC) potential [10] to uniquely specify phonons in the system. Second, we

deterministically evaluate the phononic response of the system through its atoms in the bulk and surfaces of fcc Cu(100) in an attempt to address the inherent, above-stated weakness of MD. Specifically, we simulate the atomic displacements as a function of time directly without the need to use velocity-velocity correlated spectral densities. This approach provides further insight into the dispersion of phonons in the lattice. The results are then compared with the literature.

### 7.1.1 Impulse-oscillation approach

Vibrations in real crystals are constrained by the boundaries and by the type and distribution of particles within the lattice [11]. These constraints give rise to densities of vibrational energy states for which a continuous spectral density distribution function,  $g(\omega)$ , can be found. The extent of oscillation coupling depends on wave energy ( $E$ ) and momentum ( $\vec{p}$ ), which are functions of wave frequency ( $\omega$ ) and wavelength ( $\lambda$ ) respectively.

In this work, we suggest an elastic strain based phonon model that can be simulated directly using MD, with strains constrained to be within the limits of Hooke's Law [11]. The calculation of wave dispersion through the bulk and surfaces is then possible, accepting that elasticity in a crystal is direction dependent. This model relies on coupled masses in an elastic medium. This approach has the advantage that it can model various scenarios with relative ease. The simplest case involves homogeneous masses, such as Cu atoms in a pure crystal. A more advanced system involves heterogeneous masses, such a doping atom in an otherwise homogeneous lattice. Also, the effect of a vacancies can readily be evaluated by direct removal of an atom to a specific point relative to the lattice to emulate Frenkel or Schottky vacancies. In this article, only the first case is considered, although the VSV software is capable of both scenarios.

Figure 7.1 shows the mass-spring oscillators models used in the simulations, with the initial impact applied on the face atom on the (100) plane. Here, 'impact' is in the mechanical,

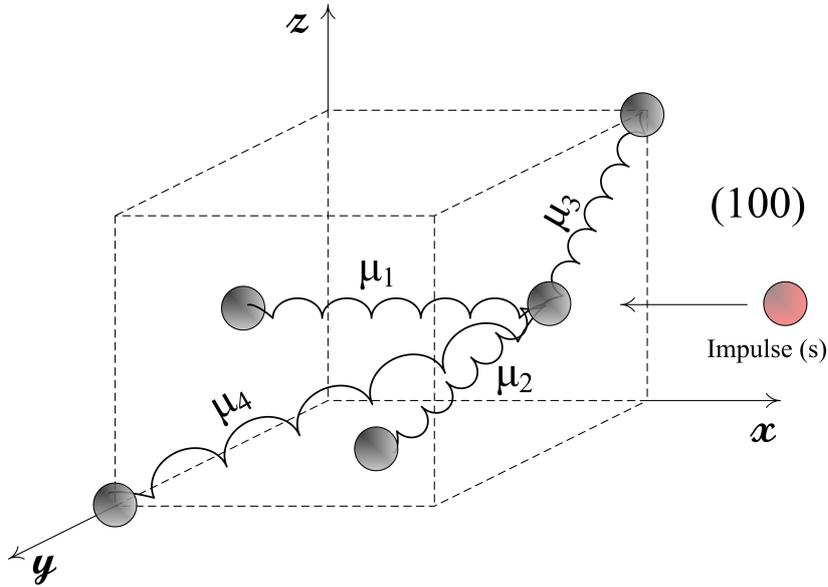


Fig. 7.1 Homogeneous mass-spring analogy of fcc structure using direction dependent spring constants.

hard-ball atom sense. The impulse represents a soft impact of the highlighted atom onto the structure. Four necessary assumptions are made here. First, the impacting atom is identical to the structure atoms (in this case Cu). Second, after the impact the impacting atom decouples and travels away from the structure. Third, the impacting kinetic energy causes deformation of the structure within the Hooke's law limit. Lastly, collision is elastic, i.e. the kinetic energy is conserved. In Figure 7.1, only the unrepeated springs are shown for the sake of clarity. In reality, the interplay of attractive-repulsive forces depending on interatomic distance in the SC potential implies that no actual physical atom contacts occur. Our simulations show that the impacting adatom does not get closer than  $2.0 \text{ \AA}$ . This approach allows the evaluation of wave propagation through the lattice and forms the basis of this present work. Applying different initial conditions at the impacting atom opens up the study of a wide variety of structures through their elastic behaviors.

Figure 7.2 depicts the mass-spring model along the  $x$ -direction. The elastic wave behavior ( $e^{i\vec{k}\cdot\vec{r}}$ ) at atom position ( $\vec{r}$ ) and wave vector ( $\vec{k}$ ) is complex due to instantaneous superimpositions of contributions from all directions. However, a deterministic simulation based

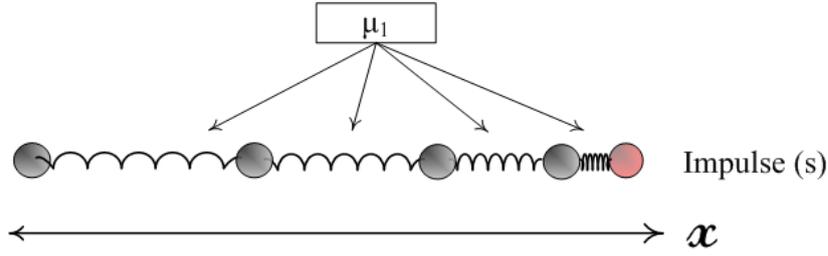


Fig. 7.2 Mass-spring analogy of fcc structure along the  $x$  axis.

on such a model can lead to a better model of phonon coupling in the bulk crystal and consequently improve understanding of effects of phonons in the lattice such as thermal properties [12, 13]. The clusters considered are homogeneous i.e. monoatomic, without any presumption of transverse or longitudinal waves,  $u_s$ , w.r.t the impulsing direction. For such a lattice comprised of atoms of mass  $m$ , if the atomic force constant of the lattice is  $\mu$  and  $k$  is the wavevector, then the dispersion relation,  $\omega(k)$ , can be written

$$\omega^2 = (4\mu/m) \sin^2(ka/2), \quad (7.1)$$

in terms of a spacing  $a$ . This spacing is equal to  $d$ , the interplanar distance for longitudinal waves, and the lattice parameter for transverse waves. The first Brillouin zone sufficiently describes the wave propagation in the lattice and is bounded by  $k=\pm\pi/a$  [11]. In our initial simulations, the effective forces were observed to influence atoms that are spread out over a distance several tens of lattice parameters. This supports observations that are made in real experiments and severely limits the utility of the cut off distance ( $r_{\text{cut}}$ ) concept typically applied in simulations to reduce computational complexity. For this reason, our calculations do not implement  $r_{\text{cut}}$ , and the nanosize of the clusters further precludes it. By considering the collective contributions of these long range forces over the first Brillouin zone, for the  $p$  nearest planes the interplanar force constant,  $\mu_p$  can readily be shown to be

$$\mu_p = -\frac{ma}{2\pi} \int_{-\pi/a}^{+\pi/a} \omega_k^2 \cos(pka) dk. \quad (7.2)$$

## 7.2 The simulation model

Koleske et al [1] have investigated surface phonon dispersion on fcc surfaces using the Lennard-Jones (LJ) ( $m$ - $n$ ) potential for simplicity. LJ potentials are known for their inadequacy to describe metallic systems [14]. This article uses the SC potential that is considered more representative of metallic systems. The total SC potential energy is given by Equations (3.10) and (3.11).

Table 7.1 shows the potential and simulation parameters.

Table 7.1 The SC simulation parameters used in the simulation test beds for Cu.

| Parameter            |              |
|----------------------|--------------|
| $\epsilon$           | 0.012382 eV  |
| $m, n, c$            | 6, 9, 39.432 |
| $\delta t$           | 91.75 fs     |
| $t_{start}, t_{end}$ | 0, 250 ps    |
| $T$                  | 0 K (bulk)   |

### 7.2.1 Simulation conditions and delimitation

The use of velocity based temperature modeling within energy equipartition theory (EPT) implies that the atomic vibrational amplitudes, measured as displacements from their equilibrium lattice positions, are smaller at low temperatures. The deformations can therefore be kept to within the low-strain region where lattice thermal conductivity can be explained using harmonic oscillators within Debye theory [14, 15]. Simulating at 0K while assuming only external phonon sources is advantageous because thermally generated statistical noise is precluded from the results. In EPT, expectation particle speed  $v_j$  and temperature  $T$  are related by

$$v_j = \left( \frac{k_B T}{m} \right)^{1/2} = \left( \frac{2E_{kin}}{N_f m} \right)^{1/2}, \quad (7.3)$$

where  $E_{kin}$  is the total kinetic energy of all the atoms, and  $N_f$  are the available degrees of freedom. VSV assumes only irrotational, translation motion, hence  $N_f=N$ . Simulations should ultimately provide insights into structures at temperatures much higher than 0K, thus making it necessary to equilibrate the initial simulation temperature to the ‘most probable’ mean isotherm for the entire lattice. For simulations at initial temperature  $T_{iso} \neq 0$ , the VSV code redistributes atomic velocities based on Maxwell-Boltzmann (MB) theory such that with all velocities reconsidered, the overall structure is at temperature  $T_{iso}$  with acceptable tolerance. This is done by introducing the randomized exponential factor  $f$  [16–19]:

$$f = (k_B T / m)^{1/2},$$

where  $k_B$  is Boltzmann constant. For 0K simulations  $e^f=1$ . At higher temperatures, anharmonic effects become important, particularly within the bulk. The simulations here are kept below the melting point i.e. within the high temperature limits of Debye theory [11, 14]. However, the onset of crystal melting is detectable by increased disorder in the atomic spacings, such as a rapidly changing variance in the calculated lattice parameter in consecutive time steps. The total potential energy and the kinetic energies of all atoms in the configuration were monitored to guarantee that the collisions are elastic. The main simulation theme is then to vary three parameters singly or in combination i.e.

- i) impacting adatom energy (i.e. velocity or EPT temperature)
- ii) impacted surface orientation
- iii) impacting power i.e. energy flux (impacts) per unit time

Therefore, the simulation highlights a number of real lattice behaviours, such as the thermal properties of given surface orientations, ablation and others. For instance, it is possible to estimate the scattered atom flux in a specified direction or onto plane in response to an ablating energy modeled using kinetic energy of the adatom stream. Knowledge of scattering

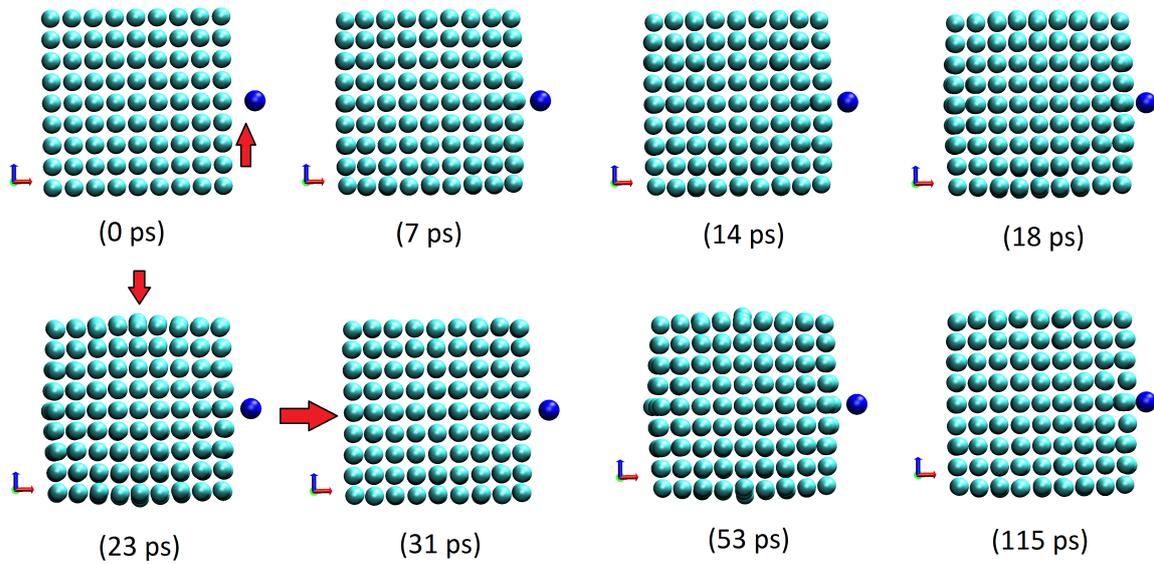


Fig. 7.3 Time-lapse views in  $[100]$  equivalent directions showing passage of impact-generated wave through the lattice generated from actual simulation. The adatom is highlighted in blue.

flux and direction can allow studies of layering by ablated atoms onto a secondary surface oriented on a specific plane.

Figure 7.3 shows the rendered, time-lapse images using data from an actual simulation. The view shown is in  $[100]$  equivalent directions. The arrows show the points where amplitude variations are readily discerned. The phase of amplitude oscillation by particle displacement at the points shown indicate that a real wave actually passes through the lattice in response to the impact.

The suggested method of impact generated phonons has a number of possible limitations. The first can arise when the low-strain requirement is violated under high impacting energy flux. Therefore, it is important to avoid large vibrational amplitudes in the initial conditions which would propagate through the lattice from the impact site. In accordance with EPT, the impacting atoms are implicitly at higher temperatures than the main lattice by virtue their larger initial velocities. Impact therefore involves energy transfer into the main lattice. Hence the temperature limits should be known *a priori* before assigning the velocities, or the velocities carefully scaled to maintain an isotherm. These considerations potentially increase

the complexity of the simulation. This has been avoided here by using energies that are equivalent to temperatures that are much lower than the Debye temperature,  $T_D = \omega_D \hbar / k_B$ , where  $\omega_D$  is the Debye frequency for the atom.

In a monoatomic lattice a density of state function,  $g(\omega)$  describes all the degrees of freedom up to  $\omega_D$ , and the frequencies are in the acoustic region ( $\sim 10^{13}$ /s). However, real lattices can have impurity atoms, which perturb the oscillations and can introduce additional modes at optical frequencies, giving rise to an energy bandgaps which depend on the direction of the most impurity atoms i.e. a sensitivity to momentum (direction),  $\bar{k}$ . Lastly, all bounding surfaces have clear termination orders which present discontinuities. They are treated here as though there is no termination of atoms i.e. an infinitely large lattice where the energy dissipation is gradual with no reflections into the bulk from the surfaces.

### 7.2.2 Results and discussions

Simulation of Equation (3.10) in VSV generates a large number of  $(t, \bar{r}, \bar{v})$  trajectory points for the collection of atoms, making visualization of the output necessary. The output data were converted to the “.xyz” molecule format using an Excel macro, described in Appendix A.4.1. Thus, instantaneous time frame and animations can be followed using Visual Molecular Dynamics (VMD) program [20].

### 7.2.3 Bond-length oscillations

Figure 7.4 shows the relative position of two copper atoms, typical in the bond length versus time simulation. Here, they are near neighbors on the face diagonal with their centers separated by 2.5526 Å in Cu at equilibrium and 0K. The adatom imparts energy to the main lattice, which oscillates in response.

Figure 7.5 shows the output of the bond length simulation. Figure 7.5a shows the bond length versus time behavior of the impacting atom and the atom around which the impact

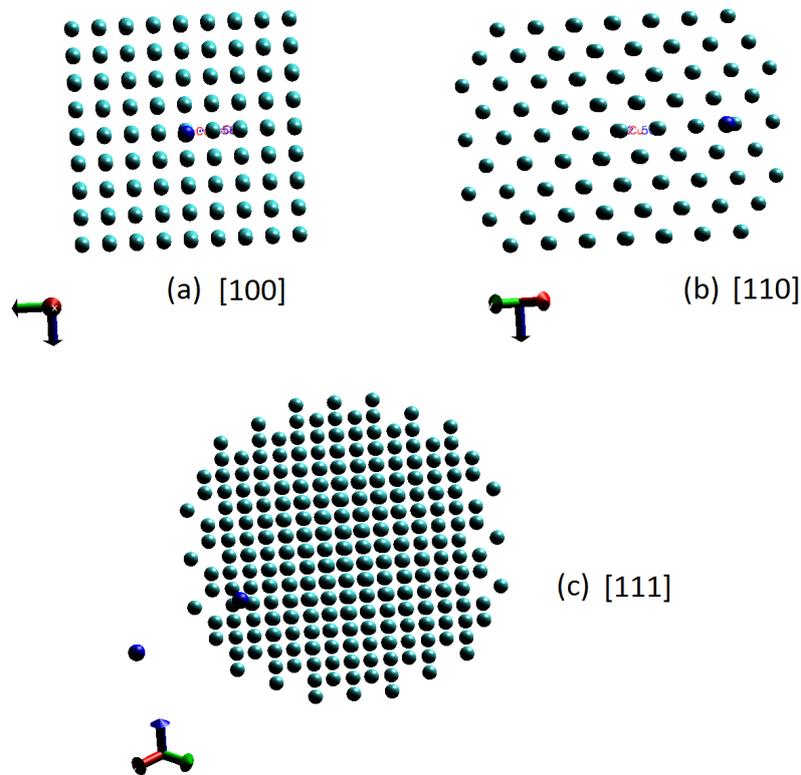
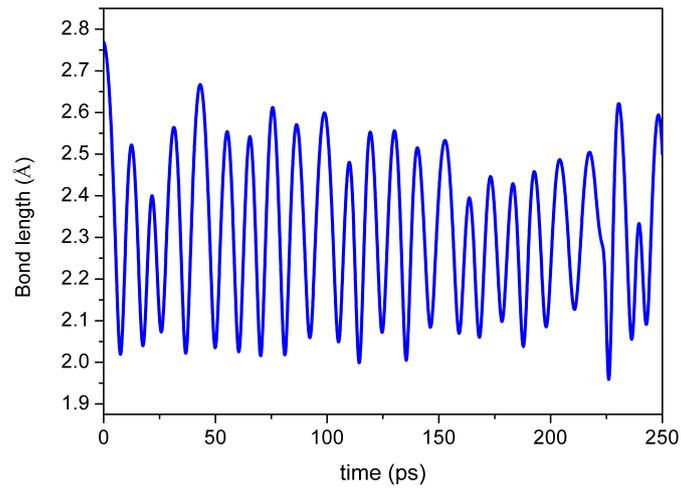


Fig. 7.4 Views along [100], [110] and [111] for the time and phase simulations below. The Cu adatom is highlighted in blue.

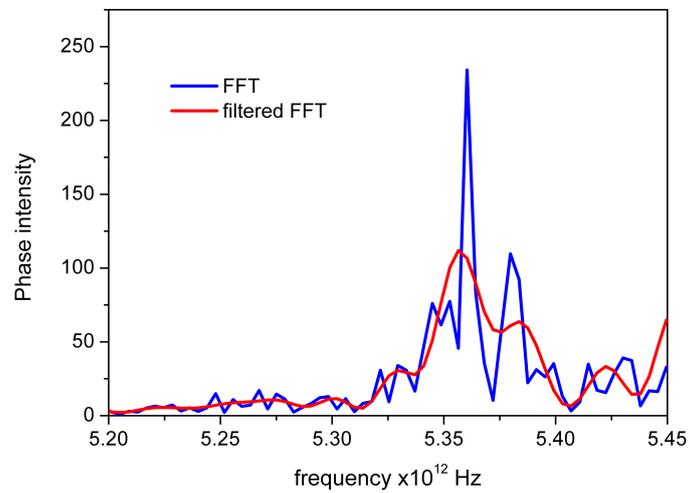
occurs. Figure 7.5b shows the discrete fast fourier transform (FFT) calculated from the amplitude response using MATLAB. Fig.7.7 plots the results of the same calculations deeper into the lattice, and using a face atom and its near neighbor i.e. corner atom. The smoothed FFT curves were obtained using FFT filtering with 25% cutoff.

Figure 7.6 shows the output of the bond length simulation in time and frequency within the bulk and along the [100] direction. Figure 7.6a shows that the bond length follows a complex, oscillatory pattern. This is expected from coupling of adjacent mass-spring oscillations, as suggested by Figure 7.1. This could explain the presence of the smaller peaks in the FFT plot.

The bond length calculated here is analogous to the spring labeled as  $\mu_3$  in Figure 7.1. Comparison of the smoothed FFT peaks in Figure 7.5b and Figure 7.7b indicates a difference in peaking frequency. Figure 7.7b shows that the peak at the point of impact is  $\approx 5.37$  THz

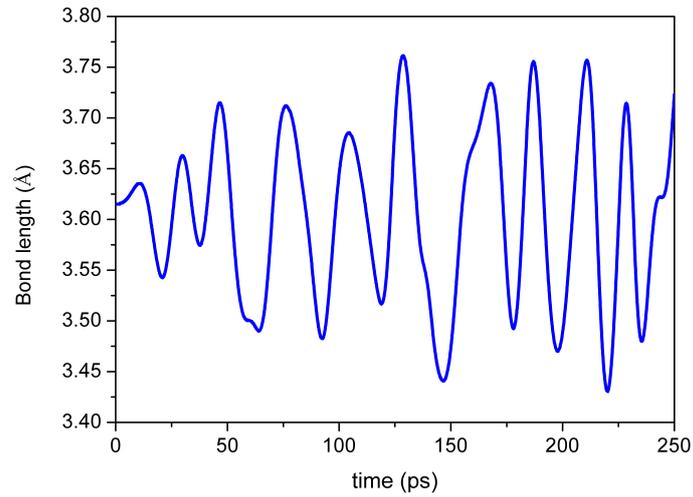


(a) Amplitude-time response

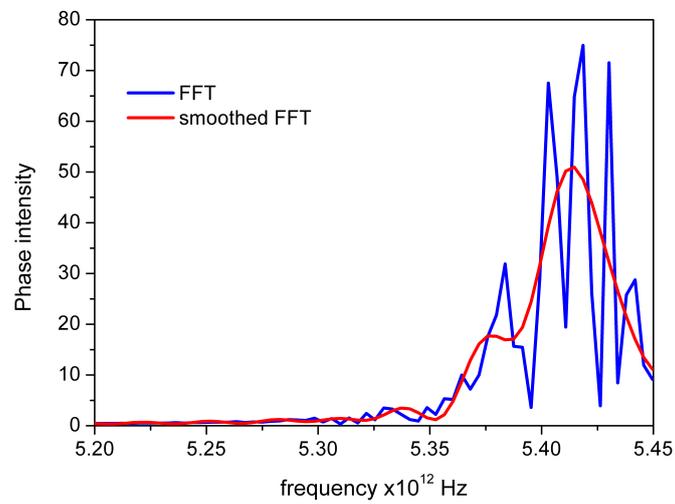


(b) Phase-frequency response.

Fig. 7.5 Simulated time and phase/frequency spectrum of bond length at the point of impact.

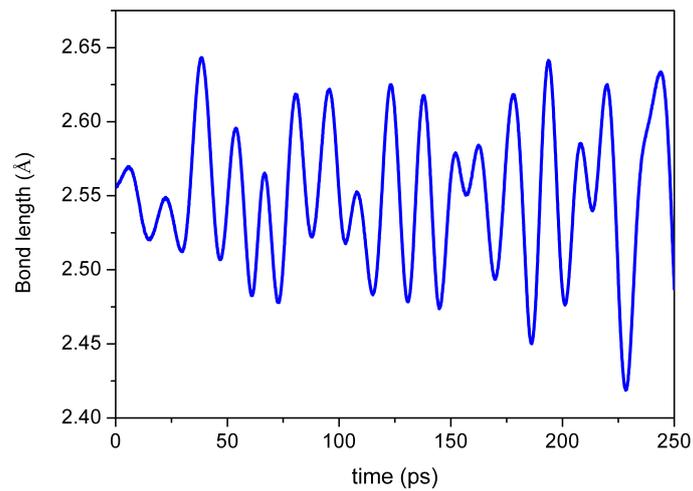


(a) Amplitude-time response

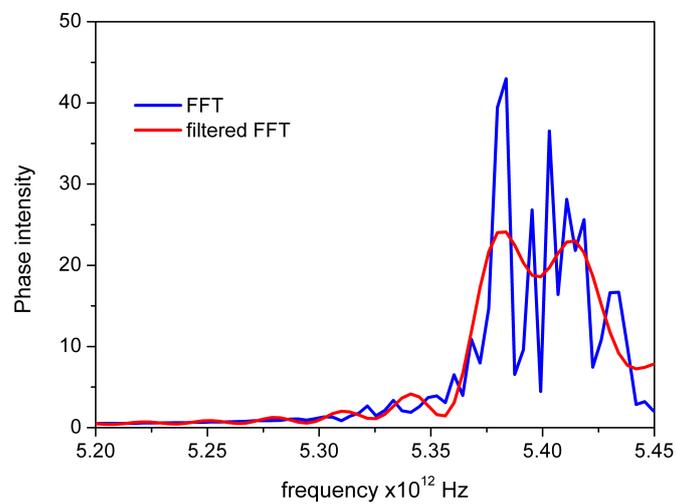


(b) Phase-frequency response.

Fig. 7.6 Simulated amplitude and phase variations in bond length along [100] deeper in the lattice, i.e. within the bulk



(a) Amplitude-time response



(b) Phase-frequency response.

Fig. 7.7 Simulated time and phase variations in bond length deeper in the lattice along [110].

( $\times 10^{12}$  Hz), against  $\approx 5.40$  THz deeper into the lattice, with an average of  $\approx 5.38$  THz. The smoothed curves also appear to suggest at least three different but closely related frequencies of oscillation over the plotted range. In the spring-mass model, this suggests at least three direction sensitive spring constants which are similar in magnitude, but account for different angular frequencies of oscillation.

The angular frequency shown in the FFT plots correspond to  $\omega \approx 3.4 \times 10^{13}$ /s, or phonon energy  $\hbar\omega \approx 22.3$  meV. This phonon energy corresponds to the experimental measurements reported in [21]. If this energy is converted purely to heat then the expected temperature rise around the two atoms is  $\Delta T = \hbar\omega/k_B \approx 259.1$ K. This temperature rise does not represent the overall temperature rise in the lattice, nor the steady state in the neighborhood of the two atoms. From monoatomic lattices of force constant  $\mu$  having with mass distributions  $m$  separated by lattice constant  $a$ , the maximum vibrational frequency  $\omega_m$  is

$$\omega_m = 2\sqrt{\frac{\mu}{m}}, \quad (7.4)$$

and occurs at  $k = (\pi/a)$  in Equation (7.1). An estimate with  $\omega_m \approx 3.4 \times 10^{13}$ /s and  $m = 63.5$  amu gives  $\mu \approx 30.8$  N/m. The literature value [11, 22] is 35.32 N/m at 296K.

#### 7.2.4 Wave propagation and the elastic constants

In the Hooke's law approximation, the moduli of elasticity  $C_{\alpha\beta}$  for a cubic crystal are directly related to the various strain components ( $e_{ij}$ ). Symmetry in the cubic structure requires three, 4-fold rotation axes. The cube is invariant under these rotations. This vastly reduces the number of elastic moduli to three. The elastic energy density can thus be written [11]

$$U = \frac{1}{2}C_{11}(e_{xx}^2 + e_{yy}^2 + e_{zz}^2) + \frac{1}{2}C_{12}(e_{yy}e_{zz} + e_{zz}e_{xx} + e_{xx}e_{yy}) + \frac{1}{2}C_{44}(e_{yz}^2 + e_{zx}^2 + e_{xy}^2), \quad (7.5)$$

such that all independent stress components  $X_a$ ,  $Y_b$  and  $Z_c$  applied along the axes  $a$ ,  $b$  and  $c \in \{x, y, z\}$  are completely defined. In summary, these elastic strain components can be calculated from only four moduli, as shown in Table 7.2.

Table 7.2 Summary of elastic strain components for a cubic crystal.

| component | $e_{xx}$ | $e_{yy}$ | $e_{zz}$ | $e_{yz}$ | $e_{zx}$ | $e_{xy}$ |
|-----------|----------|----------|----------|----------|----------|----------|
| $X_x$     | $C_{11}$ | $C_{12}$ | $C_{12}$ | 0        | 0        | 0        |
| $Y_y$     | $C_{12}$ | $C_{11}$ | $C_{12}$ | 0        | 0        | 0        |
| $Z_z$     | $C_{12}$ | $C_{12}$ | $C_{11}$ | 0        | 0        | 0        |
| $Y_z$     | 0        | 0        | 0        | $C_{44}$ | 0        | 0        |
| $Z_x$     | 0        | 0        | 0        | 0        | $C_{44}$ | 0        |
| $X_y$     | 0        | 0        | 0        | 0        | 0        | $C_{44}$ |

In a uniformly expanding cube the dilations along  $x$ ,  $y$  and  $z$  are equal. The energy density is then

$$U = \frac{1}{6}(C_{11} + 2C_{12})\delta^2 = \frac{1}{2}B\delta^2 \quad (7.6)$$

where  $\delta/3=e_{xx}=e_{yy}=e_{zz}$ , and  $B$  is the bulk modulus or energy per unit volume,

$$B = \frac{1}{3}(C_{11} + 2C_{12}). \quad (7.7)$$

For a cubic crystal, the equation of motion for displacement in the  $x$  direction and medium density  $\rho$  is

$$\rho \frac{\partial^2 u}{\partial t^2} = \frac{\partial X_x}{\partial x} + \frac{\partial X_y}{\partial y} + \frac{\partial X_z}{\partial z}. \quad (7.8)$$

Table 7.2 allows Equation (7.8) to be written

$$\rho \frac{\partial^2 u}{\partial t^2} = C_{11} \frac{\partial e_{xx}}{\partial x} + C_{12} \left( \frac{\partial e_{yy}}{\partial x} + \frac{\partial e_{zz}}{\partial x} \right) + C_{44} \left( \frac{\partial e_{xy}}{\partial y} + \frac{\partial e_{zx}}{\partial z} \right), \quad (7.9)$$

or

$$\rho \frac{\partial^2 u}{\partial t^2} = C_{11} \frac{\partial^2 u}{\partial x^2} + C_{44} \left( \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) + (C_{12} + C_{44}) \left( \frac{\partial^2 v}{\partial x \partial y} + \frac{\partial^2 w}{\partial x \partial z} \right), \quad (7.10)$$

where  $u$ ,  $v$  and  $w$  are the components of the deformation. Equation (7.10) superimposes three independent displacements. The equation shows that the waves are constrained to three modes of vibration, namely

1. along  $x$ , i.e. the [100] direction with waves of the form

$$u(x,t) = u_0 e^{i(kx - \omega t)}, \quad (7.11)$$

where  $u_0$  is the amplitude and  $k = 2\pi/\lambda$  is the wave vector. This solution can describe both transverse, longitudinal and shear waves.

2. simultaneously along  $x$  and  $y$  only, such as the [110] direction

$$w(x,t) = w_0 e^{i(k_x x + k_y y - \omega t)}, \quad (7.12)$$

where  $w_0$  is amplitude.

3. simultaneously in non-zero  $x$ ,  $y$  and  $z$  directions, such as the [111] direction,

$$v(x,t) = v_0 e^{i(k_x x + k_y y + k_z z - \omega t)}, \quad (7.13)$$

where  $v_0$  is amplitude.

In the [100] direction, substitution of Equation (7.11) into Equation (7.10) gives for a longitudinal wave

$$\omega^2 \rho = C_{11} k^2, \quad (7.14)$$

and

$$\omega^2 \rho = C_{44} k^2, \quad (7.15)$$

for a transverse or shear wave. The wave vector is along the  $x$  edge of the cube and particle displacement is along the  $y$  direction. For waves propagating along face diagonal,  $C_{11}$ ,  $C_{12}$

and  $C_{44}$  can be found using Equation (7.12) in the special case of the [110] direction that the transverse wave arises when

$$\omega^2 \rho = \frac{1}{2}(C_{11} + C_{12} + 2C_{44})k^2. \quad (7.16)$$

The shear wave arises when

$$\omega^2 \rho = \frac{1}{2}(C_{11} - C_{12})k^2. \quad (7.17)$$

#### 7.2.4.1 Estimating C11 and C44

The condition that  $\omega(k)=\omega_m$  when  $k=\pi/a$  in Equation (7.14) gives

$$C_{11} = \left(\frac{\omega_m a}{\pi}\right)^2 \rho = \left(\frac{2\omega_m}{\pi}\right)^2 \frac{M}{a}, \quad (7.18)$$

with  $a=b\sqrt{2}$ .

A calculation based on the data of Figure 7.5b, which represents the [100] direction, gives  $C_{11} = 139.2 \pm 1.4$  GPa using  $f_m = 5.44$  THz. Experimental measurements give  $C_{11}$  in the range 176.2 to 168.4 GPa at 0K and 300K respectively [11, 22]. The average bond length from the data of Figure 7.7a was found to be  $b = 2.5487\text{\AA}$  for diagonal measurements, i.e. [110] and [111] directions. Only bulk densities, which are densities calculated using length parameters such as  $a$  or  $b$  deeper in the lattice were used, i.e.

$$\rho = \frac{4M}{a^3} = \frac{M\sqrt{2}}{b^3} = 9058.3 \text{ kg/m}^3, \quad (7.19)$$

with  $M = 63.5$  a.m.u. The error margins in the bond length ( $a$ ) and material density ( $\rho$ ) can be estimated by applying variational calculus to Equation (7.18) and Equation (7.19). It can

readily be shown that for maximum calculated uncertainties:

$$\frac{\Delta a}{a} = \left( \frac{\Delta C_{11}}{C_{11}} + 2 \frac{\Delta \omega_m}{\omega_m} \right), \quad (7.20)$$

and

$$\frac{\Delta \rho}{\rho} = \frac{1}{3} \frac{\Delta a}{a}. \quad (7.21)$$

The estimate of  $\Delta \omega_m$  can be made by discretizing a phase-frequency response curve. In the specific case of Figure 7.6b, for example, the angular frequencies range from 5.35 THz to 5.45 THz, with a peak at 5.40 THz. Thus, these three frequencies give a standard deviation of  $\Delta \omega_m = 0.0314$  THz. Applying Equation (7.20) on the stated  $C_{11} = 139.2 \pm 1.4$  GPa and  $\omega_m = 5.40 \pm 0.0314$  THz gives  $\Delta a = \pm 0.0553$  Å. By a similar token, applying Equation (7.21) leads to  $\Delta \rho = \pm 65.5$  kg/m<sup>3</sup>. The estimated percent uncertainties in the bond lengths and density are respectively 2% and 0.7%.

Behari and Tripathi [22] reported calculated frequency spectra for Cu, Ni and Al at 0K. The results of the calculated spectra in this paper are similar. They report a broad peak in the  $g(\omega)$  versus  $f$  between 4 and 5 THz, and a single narrow peak around 7 THz for all three metals. The chosen simulation time step of 91.75 fs gives a sampling frequency of 11 THz, which limits the maximum observable frequency to 5.6 THz in accordance with the Nyquist sampling theorem. Therefore, for any peaks between 7 to 10 THz, the sampling frequency needed is at least 20 THz, which requires a maximum simulation step of 50 fs. The range of maximum spectral frequencies were not known *a priori*. The parameters in Table 7.1 were based on other reduced (dimensionless) length, mass and energy parameters and gave the 91.75 fs step as practicable. Subsequent simulations for the spectral response will use 50 fs.

#### 7.2.4.2 Expected temperature rise

The ‘impact’ with the lattice causes vibrations that propagate as both bulk and surface phonons, which propagate differently in the three symmetry directions [23, 24]. The output data point  $(t, \bar{r}, \bar{v})$  allows instantaneous temperature at any atom location to be calculated according to Equation (7.3). Therefore, a three dimensional map of the temperature distribution in the lattice can be generated at each time step for any arbitrary plane. This map visualizes the temperature variation in response to the impacting adatoms. However, temperature cannot simply be calculated using one atom, but an entire, three dimensional neighborhood of the atom.

### 7.3 Conclusions

This article successfully employs an evolving code to carry out deterministic simulations based on the Sutton-Chen embedded atom potential. Therefore, it provides a novel solution to the inherent weakness of MD by relating for the first time atomic displacements to spectral densities and vibrational modes in a simulation. Specifically, we investigate the surface and bulk interactions in small clusters of up to 373 fcc atoms. The study highlights some crystal formation dynamics and growth mechanisms through the capture of additive atoms. The size of the fcc cluster is kept small for speed up accessing the effects of the interactions. The calculations pertain to copper atoms and are visualized in a detailed, time-based way at the level of all the atoms present. The interactions are based on the Sutton-Chen potentials with the energies controlled to keep all attractions/repulsions within the low-strain regions of the cluster. We also show that superimposed harmonic oscillations are the result of the interplay of the interatomic forces in the lattice. The main cluster was kept initially to 0K temperature, without any Maxwell-Boltzmann spread of temperature in order to isolate random thermal effects and maintain focus on wave behavior alone. The results of bond length, material

density, elasticity constants, and spectral densities are in excellent agreement with the literature. The atom configurations resulted in a cluster of overall nanosized dimensions. The simulation results suggest that the foregoing, mechanical properties of the nanocluster are identical to bigger macroscopic clusters.

These simulation results are encouraging and suggest considerable scope for future research. For instance, comparison of the dispersion relations in specific wave directions  $k$  by simulation alone is instructive. The present paper provides the requisite starting point for such an investigation. Also, the effect of different impacting energies on the lattice, which represents external heating of the lattice can immediately be simulated for the temperature dependencies of the foregoing parameters. This latter question is interesting and important from the engineering perspective of heating and cooling. Since temperature is modelled using velocity within EPT, such a research can completely map the thermal properties of specific lattice orientations and lead to a better understanding time-transient heating and cooling of nanodevices.

# References

- [1] Koleske DD, Sibener SJ. Phonons on fcc (100), (110), and (111) surfaces using Lennard-Jones potentials. *Surface Science*, 268:407–417, 1992.
- [2] Black JE. in Schommers W, von Blankenhagen P (Eds.) *Structure and Dynamics of Surfaces*. Springer, Berlin, 1986.
- [3] Kress W, de Wette FW. Surface phonons. *Springer Series in Surface Science*, 21, 1991.
- [4] Rassoulinejad Mousavi SM, Mao Y, Zhang Y. Physical contributions to the heat capacity of nickel. *Journal of Applied Physics*, 119(244304):861–871, 2016. doi: 10.1063/1.4953676.
- [5] Allen RE, Alldredge GP, de Wette FW. Studies of vibrational surface modes. I. General formulation. *Phys. Rev. B*, 4(6):1648, 1971.
- [6] Allen RE, Alldredge GP, de Wette FW. Studies of vibrational surface modes. II. Monatomic fcc crystals. *Phys. Rev. B*, 4(6):1661, 1971.
- [7] Chen ET, Barnett RN, Landman U. Crystal-melt and melt-vapor interfaces of nickel. *Phys. Rev. B*, 40(2):924–932, 1989. doi: 10.1103/PhysRevB.40.924.
- [8] Chen ET, Barnett RN, Landman U. Surface melting of Ni(110). *Phys. Rev. B*, 41(1): 439–450, 1990. doi: 10.1103/PhysRevB.41.439.

- [9] Ocaya R, Terblans JJ. Addressing the challenges of standalone multi-core simulations in molecular dynamics. *in P. Ramasami (ed.), Computational Sciences, De Gruyter*, pages 1–21, 2017. doi: 10.1515/9783110467215-001.
- [10] Ocaya R, Terblans JJ. Temperature specification in atomistic molecular dynamics and its impact on simulation efficacy. *J. Phys.: Conf. Ser*, 905(012031), 2017. doi: 10.1088/1742-6596/905/1/012031.
- [11] Kittel C. *Introduction to Solid State Physics*. John Wiley & Sons, New Jersey, 2005.
- [12] Mohapatra M, Tolpadi S. Effects of lattice dispersion and elastic anisotropy on the thermal properties of fcc metals. *Pramana*, 35(2):159–165, 1990.
- [13] Meschter PJ, Wright JW, Brooks CR, Kollie TG. Physical contributions to the heat capacity of nickel. *Journal of Physics and Chemistry of Solids*, 42(9):861–871, 1981.
- [14] Schommers W. *in Schommers W, von Blankenhagen P (Eds.) Structure and Dynamics of Surfaces*. Springer, Berlin, 1986.
- [15] Pathak LP, Rai RC, Hemkar MP. Lattice vibrations and debye temperatures of transition metals. *Journal of the Physical Society of Japan*, 44(6):1834–1838, 1978.
- [16] Knuth D. *The art of computer programming, seminumerical algorithms*. Addison-Wesley, 1997.
- [17] Box G, Muller M. A note on the operation of random normal deviates. *Ann. Math Stat.*, 29:610–611, 1958.
- [18] Marsaglia G. The role of surface and interface structure in crystal growth. *Proc. Nat. Acad. Sci.*, 61:25–28, 1968.
- [19] R O Ocaya and J J Terblans. Coding considerations for standalone molecular dynamics simulations of atomistic structures. *Journal of Physics: Conference Series*, 905(1):

- 012018, 2017. doi: 10.1088/1742-6596/905/1/012018. URL <http://stacks.iop.org/1742-6596/905/i=1/a=012018>.
- [20] Theoretical and Computational Biophysics Group. Visual Molecular Dynamics, 2017. URL [http://www.ks.uiuc.edu/Research/vmd/allversions/what\\_is\\_vmd.html](http://www.ks.uiuc.edu/Research/vmd/allversions/what_is_vmd.html). Accessed: 2017-11-02.
- [21] Ditlevsen Peter D, Nørmskov Jens K. Vibrational properties of aluminum, nickel and copper surfaces. *Surface Science*, 254:261–274, 1991. doi: 10.1.1.718.7356.
- [22] Behari J, Tripathi BB. Frequency spectra and heat capacities of copper, nickel and aluminium. *Aust. J. Phys.*, 23:311–318, 1969. URL <http://adsabs.harvard.edu/full/1970AuJPh..23..311B>.
- [23] Chopra KK, Bouarab S. Off-symmetry phonon frequencies of copper. *Physica status solidi (b)*, 125(2):449–504, 1984.
- [24] Jani AR, Gohel VB. On the phonon frequencies of copper in off symmetry directions. *Solid State Communications*, 41(5):407–411, 1982.

# Chapter 8

## Post-MD simulation visualization in VSV

### 8.1 Introduction

In recent years, nanosized metallic clusters have become important to both experimental and computational research. Currently, considerable challenges remain. Empirical research continues to face size restrictions associated with synthesis and physical probing techniques, while simulation faces problems associated with computation, such as efficacy of the mathematical model for simulation, hardware speed and accuracy difficulties and so on [1–4]. The correct visualization of simulated output is as important as the accuracy of the simulation itself. This can be challenging in the case of simulations that are done on own developed software. This hurdle is also experienced in empirical situations where vast intermediate data are generated from a number of sources, that must be combined into a visual form, for instance in image processing [5, 6]. In a previous article [7], we describe VSV for standard molecular dynamics, based on the Sutton-Chen embedded atom model to simulate metallic fcc structures. VSV output data is massive but contains a wealth of different information that requires post-processing to interpret meaningfully. Although computer simulations have evolved, the problems of visualization still remain in some form or other [8, 9].

## 8.2 Motivation and significance

We have presented a collective method to process the output of the Sutton-Chen (SC) potential based VSV simulations into various intermediate outputs for interpretation. VSV simulates homogeneous fcc atomic clusters using a low strain phonon model. The continuous spectral density vibrational are put into the context of Debye theory [10], where thermal conductivity is the net effect of harmonic oscillators with  $N$  degrees of freedom [11, 12]. The method comprises a Visual Basic macro within Microsoft Excel, and MATLAB codes. The macro extracts vibrating surfaces using user-specified parameters. The MATLAB codes calculate the Fast-Fourier Transform (FFT) of bond length versus time variations and also plots instantaneous time thermograms for the extracted surfaces. Thus, 2D-surface or 3D-subvolume thermal maps are easily rendered to visually describe passage of the wave, frequencies present are readily quantified, and instantaneous slices of the structure taken. The complete codes are maintained in a public GitHub repository [13] for evaluation by the interested reader. The included data is from an actual VSV simulation of 2281 Cu atoms that is initially perturbed by applying a low amplitude compressive disturbance to one atom on the [100] face. Real lattices are expected to oscillate as a direct consequence of the interplay between interatomic attractions and repulsions at displacements relative to the potential energy minimum. This atomic scale behavior accounts for elasticity at all scales of the material. The novelty of the present and ongoing work is its suggestion of a deterministic, approximation free ab initio approach to investigate atomic clusters by a perturbation-oscillation model that has been found to account for bulk and surface phonons that adds a useful tool to the growing MD arsenal for standalone simulation and visualization.

The impulse-oscillation approach used in this work was shown in Figure 7.2. The resultant elastic wave has wave vector  $\vec{k}$ , and amplitude-phase given by  $e^{i\vec{k}\cdot\vec{r}}$  at atom position  $\vec{r}$ . The wave dispersion in a homogeneous lattice of mass distribution  $m$ , lattice constant  $a$  and atomic force constant  $\mu$  was described by Equation (7.1) [10]. At the first Brillouin zone

boundary  $k=\pm\pi/a$ , hence the vibrational frequency is at maximum. VSV does not employ cut-off distance approximations and is suited to nanosized cluster simulations, accepting that interactions between atoms separated up to 20 lattice parameters cannot be ignored.

## 8.3 Description of additional applets

The total SC potential energy for fcc structures is covered in detail elsewhere [14–16]. For an ensemble of  $N$  degrees of freedom, temperature  $T$  can be expressed in terms of expectation energy as given by Equation (7.3). Our temperature modeling approach in VSV simulations contrasts the various isotherms that are used to describe temperature by requiring an *a priori* knowledge of an initiating temperature [16]. Instead, we take temperature as being due to purely phonon vibrations that can be set by imparting an initial impulse energy or total internal energy. This solves a current limitation of MD since it allows the lattice to relax dynamically to a thermal equilibrium in response to the elastic waves. Even at higher temperatures where lattice anharmonicity occurs because of violations of the low-strain criterion, the impulse-oscillation approach has been used successfully to describe effects such as melting. Our monoatomic cluster simulations with Cu have been done below the Cu Debye temperature of 347 K. The observed frequencies are acoustic i.e.  $\sim 10^{13}/s$ , and lie in the infrared region. The presence of another atom ( $m_2$ ) may give the lattice a diatomic character and introduce an optical branch [10].

### 8.3.1 Helper routines

The following sections describe the various routines and applets that assist the interpretation of data generated by the main simulation in the VSV program.

### 8.3.1.1 Highlighting adatoms for VMD

Figure 8.1 shows the results of a post-processing on VSV output which highlights the adatom for visibility in VMD [17]. All atoms, including the highlighted, are Cu atoms. The impulse energy is delivered to the highlighted atom through a small initial displacement, as shown in Figure 7.2. Table 8.1 shows the file structures of the various files that form the simulation

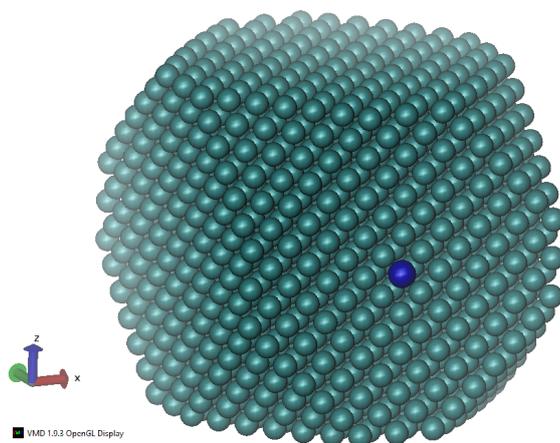


Fig. 8.1 Post-processing scheme used to highlight adatoms within VMD. The structure shown is a VMD frame showing an instantaneous structure of 2281 Cu atoms in the VSV simulation output.

chain, from input of the structure text file into VSV, to the generation of the VMD xyz input, and the extraction of the vibrating plane of atoms. The VSV input is a text file that lists all the atoms with their initial 3D component coordinates and velocities. The adatoms are placed last in the file. For instance, if only one adatom is used then the last item in the file describes that particular adatom. If 16 adatoms are used then the last 16 text file entries specify those adatoms. The post-processing step 2 has provision to automatically detect the adatoms based on the user input, and automatically highlight them for VMD, which receives the xyz file output from the step. Table 8.1 also shows the structure of the file. To highlight the adatoms within VMD we use a trick that exploits the fact that VMD uses a different color for each <element>. Thus, when the xyz file [18] is created in the case of Cu atoms, every

Table 8.1 The file formats used as primary input, intermediate and post-processed outputs in the VSV MD simulation flow.

| VSV input<br>(C-code)<br>(raw structure) | Post-process 1 output<br>(VB code)<br>(.xyz file [18], for VMD) | Post-process 2 output<br>(MATLAB code)<br>(extracted surface) |
|--|---|---|
| m <x> <y> <z> <vx> <vy> <vz>             | <number of atoms>   | <x> <z> <E>   |
| ...                                      | comment line  | ...   |
| mA <x> <y> <z> <vx> <vy> <vz>            | <element> <x> <y> <z>   |   |
|  | ...   |   |

atom detected as an adatom has its <element> property set to <Ni> i.e. nickel, which VMD displays in blue. VMD does not change the VSV simulation output, only how it is rendered.

Table 8.1 lists file types shown in the simulation stages. The VSV input file for an  $N$ -atom lattice contains  $N$  lines, and has negligible disk storage size (i.e.  $< 1$  KB). The xyz file, on the other hand, is vastly larger since it holds the system trajectories (time-step, 3D position, 3D velocities) over several integration periods per particle. For the 2281 atom Cu simulation in the demonstration, it amounts to over 24 MB. The extracted surfaces, with the format also shown in Table 8.1, represent slices of the structure at a user-specified time step, also has negligible storage size since it involves only one plane which contains considerably fewer atoms. Also, depending on the plane, certain components in the 3D positions could be suppressed, as is the case when one examines (100) equivalent planes.

Figure 8.2 is the graphical user interface of the Excel macro contained in the repository file `SurfaceDetect-demo.xlsx`. The file is preloaded with actual VSV simulation output for the structure in Figure 8.1. The integrations are done in 0.5 steps from 0 to 109.5 units, a total of 219 steps. This corresponds to  $(109.5 \times 2281 / 0.5) = 499,539$  lines in the Microsoft Excel spreadsheet, or data file size of 24 MB. We have imposed this limitation to keep within the 25 MB file upload size limit of Github. Our work was based on 459 steps (47 MB). Excel supports a maximum of 1,048,576 lines per workbook. Therefore, proceeding in 0.5 simulation steps for 2281 atoms permits a maximum simulation time of  $(1,048,576 / 2281) = 459$  whole samples. That is, for the  $\equiv 45.875$  fs sample rate used [7, 16, 19], then the

Program generates "out.xyz" trajectory frames for VMD (c) ocaaya, r.o. - 2017

Total Number of atoms = 2281

Number of adatoms (16 default) = 1

Simulation time step (dt) = 0

Simulation end time (t\*) = 0

Generate trajectory file

Surface detection

Set time step to examine surface = 100

Min. angle (deg) in cross product test = 85.0

Enter coordinates of three atoms on desired plane

|        |        |       |        |
|--------|--------|-------|--------|
| Atom 1 | 1.773  | 14.38 | -5.298 |
| Atom 2 | -5.353 | 14.31 | -8.987 |
| Atom 3 | -7.179 | 14.34 | 7.18   |

Detected parameters

Starting particle index = 1

Stopping particle index = 1

Basis vectors (u.v.w) = 0

Detect atoms on plane

Total surface energy

Fig. 8.2 Graphical user interface of the Excel macro used for file generation and surface detection.

maximum simulation time is 21.056 ps. The repository also contains the VMD compatible file out .xyz which is output in response to “Generate trajectory file”. This file is similarly limited to 221 samples, or VMD frames, for 24.7 MB disk size and 10.138 ps total simulation time.

### 8.3.1.2 Tracking atoms on vibrating planes

Figure 8.3 depicts the effect of the passage of a transverse lattice wave on the position of atoms. For instance, atom 4 normally resides on an arbitrary, undisturbed plane. The passing transverse wave energy displaces it to a new position, e.g. atom 5. The wave amplitude  $|\bar{r}_5 - \bar{r}_4|$  is assumed small to comply with the low-strain condition. To examine a given plane, it is necessary to detect all displaced atoms on the plane and to selectively extract their instantaneous trajectories. Three atoms {1,2,3} were identified on the desired plane and

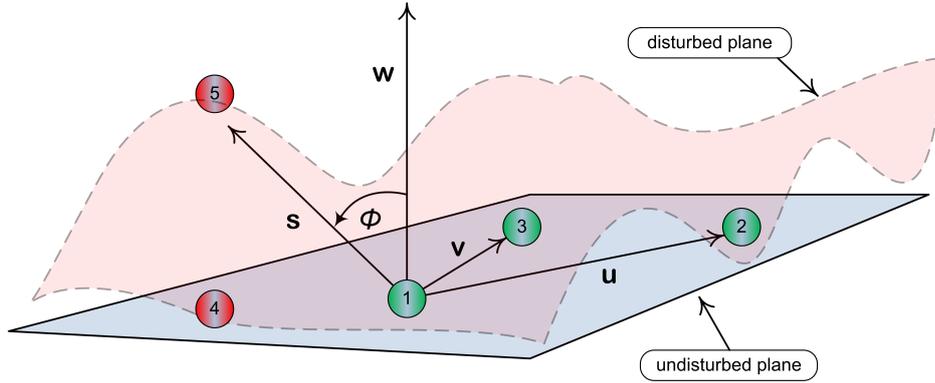


Fig. 8.3 Depiction of a superimposed, transverse impact-generated wave propagating on a planar slice of the lattice.

used to generate three vectors  $\{\bar{u}, \bar{v}, \bar{w}\}$ , such that  $\bar{w} = \bar{u} \times \bar{v}$ . Hence to identify a given displaced atom over the entire nanostructure as originating from the undisturbed plane e.g. atom 5, the orientation of its vector  $\bar{s}$  relative to  $\bar{w}$  is tested. Since  $\bar{w}$  is perpendicular to the undisturbed plane, atom 5 is confirmed as being from the undisturbed plane provided that

$$\frac{\bar{w} \cdot \bar{s}}{|\bar{w}| |\bar{s}|} \leq \cos \phi, \quad (8.1)$$

where  $\phi$  is a specifiable minimum angle. In Figure 8.3, atom 5 lies on the undisturbed plane when  $\phi = 90^\circ$ . In the present simulations  $\phi \approx 85.0^\circ$  was found to detect all displaced atoms of the original plane. The arguments also apply to longitudinal and shear waves.

### 8.3.1.3 Spectral response through bond length

We detect oscillations within the structure by selecting arbitrary adjacent atoms in the direction of interest. For instance, two atoms may be selected along the [100] direction and the variations of their separation over the simulation time extracted and used to plot a graph. Such a graph shows the net oscillation, which is a superimposition of all harmonic oscillations in the lattice due to the elastic coupling from neighboring atoms. In the low strain limit the oscillations preserve the fundamental spectral character of the lattice that

can be processed further using the Fourier transform. Figure 8.4 shows how bond lengths are calculated in VMD. In the figure, the “Save” option produces a separate vector text file

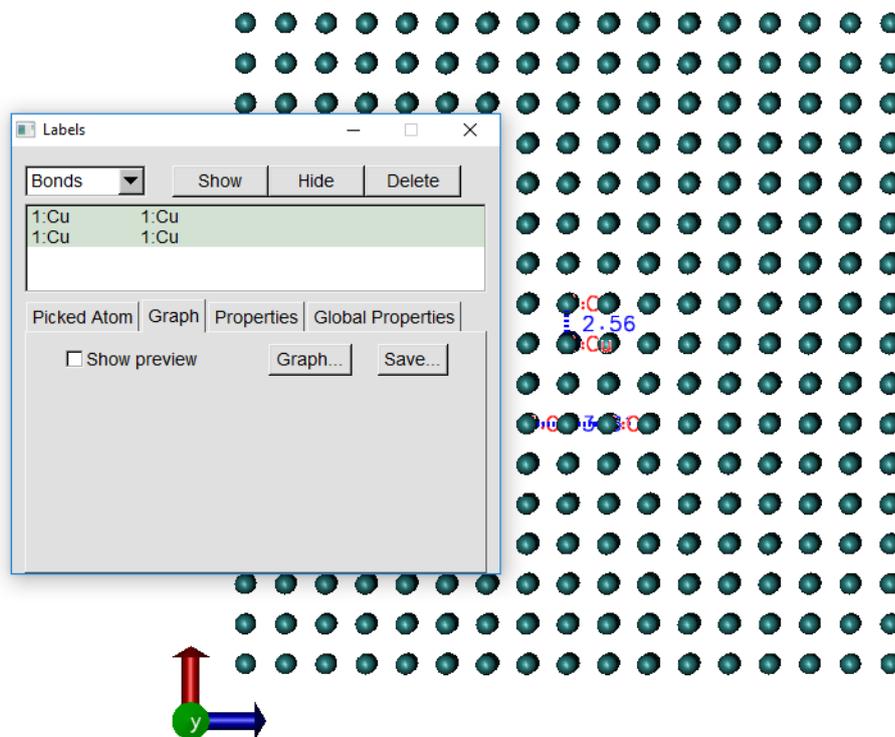
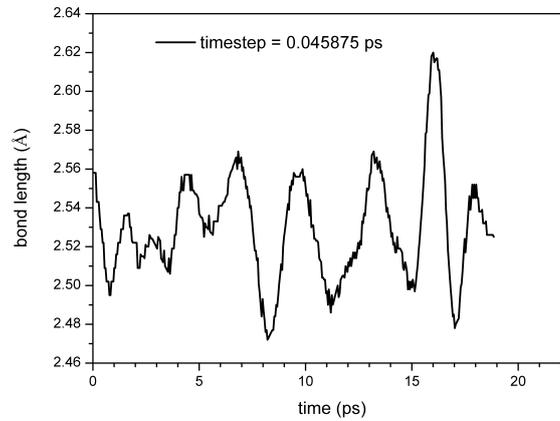
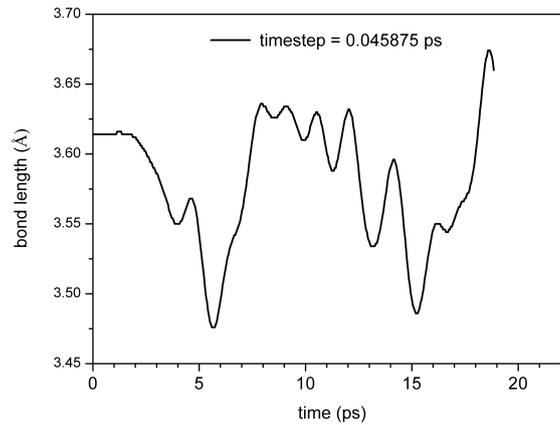


Fig. 8.4 Highlighted bond lengths in VMD showing one view of the simulated 2281 atom Cu structure after the first simulation step. The bond lengths are in Å.

with the format (timestep, var1) for each data variable over the entire simulation duration for the selected bonds. To generate a single matrix file with structure (timestep, var2, var2), which is easier to plot in other environments such as Origin, the button labeled “Graph” is used, which then generates a graphical plot within VMD. By selecting “File”, “Export to ASCII matrix” the user can then save the variables to a text file. In the case of the evaluation data, each 0.5 time step advances the simulation by 45.875 fs. Hence the text file can be further processed within Excel to convert the time step to real time. Such a file, for the bond lengths in Figure 8.4, is included as `multiplot.txt` in the repository and plotted in Origin in Figure 8.5. Figure 8.5 shows complex harmonic oscillations in bond lengths as a result of superimposition of multiple harmonic oscillations. Fast Fourier transform (FFT)



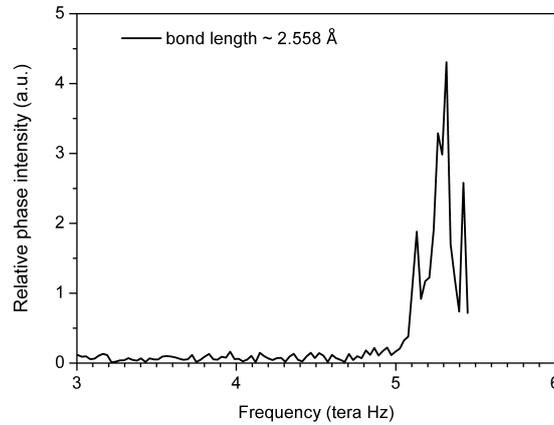
(a) [110] direction



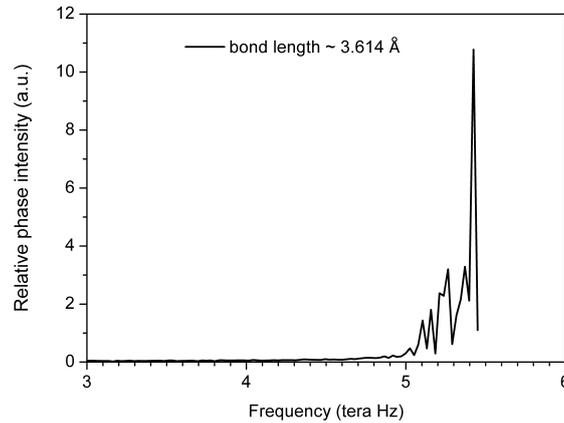
(b) [100] direction

Fig. 8.5 Simulated atom-atom bond length versus time variations in two directions.

was then used to determine vibrational mode frequencies in the bond length oscillations for the specified direction. The FFT MATLAB code, given as `FFT.m` in the repository, takes `fft_in.txt` as input and generates `fftout.txt` as output. The input and output formats are (realtime, bondlength) and (hertz frequency, intensity) respectively. The intensity indicates the relative vibrational mode amplitude in arbitrary units. Figure 8.6 shows the output of `FFT.m` on the bond lengths in Figure 8.5. The FFT input-output data for Figure 8.6b are included in the repository. The FFTs show the frequencies present and their relative strengths are direction dependent. This is expected since lattice elasticity is known to vary with



(a) along face diagonal



(b) along lattice parameter

Fig. 8.6 FFT plots of simulated bond length variations for two directions.

direction. The various elasticity dependent constants of the macro-material are calculated from this data, as shown below.

## 8.4 Illustrative example

Figure 8.7 shows the chosen simulated region in energy-momentum space, labeled A and B. Point C is the boundary of the first Brillouin zone i.e  $\pi/a=0.87$ , where strong wave reflections are expected. The straight, solid line represents the continuum lattice. The peaks in Figure

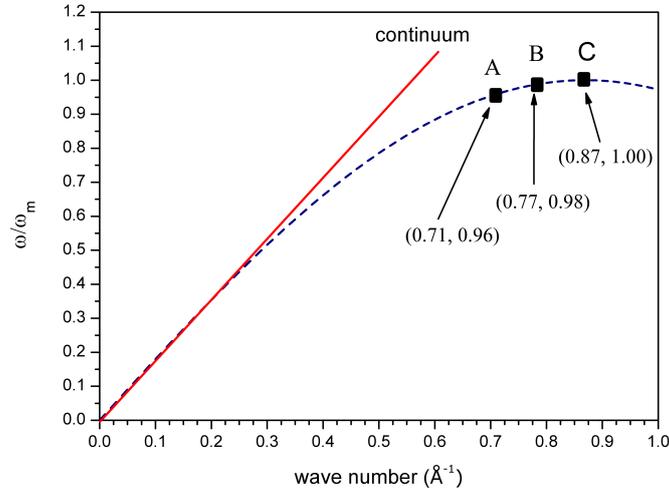


Fig. 8.7 Illustration of the chosen simulation region which lies between A and B.

8.6 match  $\hbar\omega$  energies of 20.9 – 22.6 meV, which are within the region A–B literature range [20]. Validation of the method is based on comparisons of the simulated parameters of the lattice with the known literature values. In particular, we calculate the elastic and mechanical constants, and the surface temperature behavior. The latter is expressed as 2D thermograms based on the outputs of the macro on VSV data.

### 8.4.1 Elastic constants

The cubic crystal  $[\alpha \beta \gamma]$  directions propagate normalized waves of the form:

$$u_{\alpha\beta\gamma} = e^{i(k_x\alpha + k_y\beta + k_z\gamma - \omega t)}, \quad (8.2)$$

where  $|(k_x, k_y, k_z)| = 2\pi/\lambda$ . For low strains, the bulk modulus is related to the direction dependent moduli of elasticity  $C_{\alpha\beta}$  by [10]

$$B = \frac{1}{3}(C_{11} + 2C_{12}). \quad (8.3)$$

Table 8.2 Summary of simulation results at 0.01K. The comparative literature value are from Kittel [10], Sutton & Chen [14], Behari & Tripathi [23].

| Source     | Speed<br>m/s | $E$<br>meV  | $c_{11}$ | $c_{12}$<br>GPa | $c_{44}$ | $\rho$<br>kg/m <sup>3</sup> | $\mu$<br>N/m |
|------------|--------------|-------------|----------|-----------------|----------|-----------------------------|--------------|
| Simulated  | 4,443        | 20.9 – 22.6 | 178.8    | 128.8           | 82.1     | 9058.3                      | 27.7 – 31.5  |
| Literature | 4,600        | 18 – 35     | 176.2    | 124.9           | 81.8     | 9018.0                      | 38.2         |

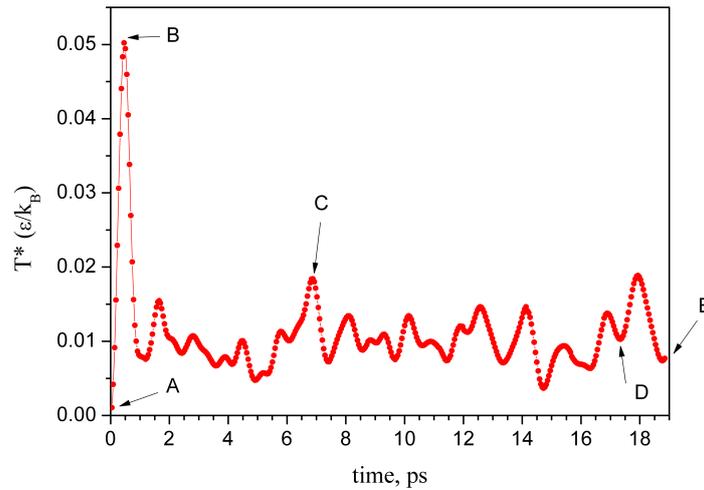
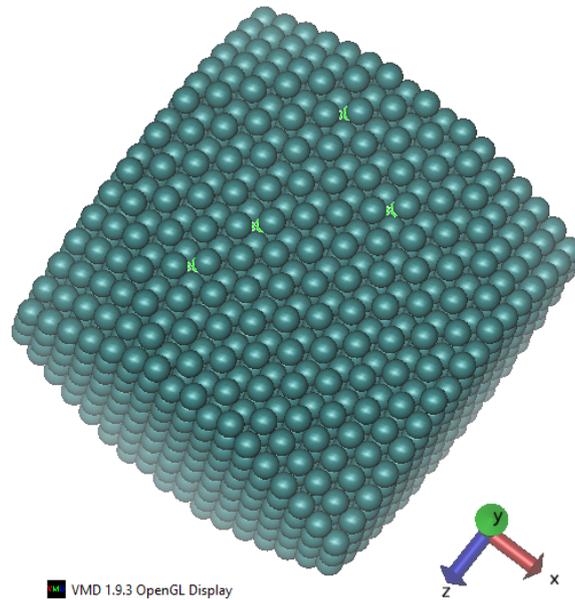


Fig. 8.8 Plot of average surface temperature in time from the moment of impact.

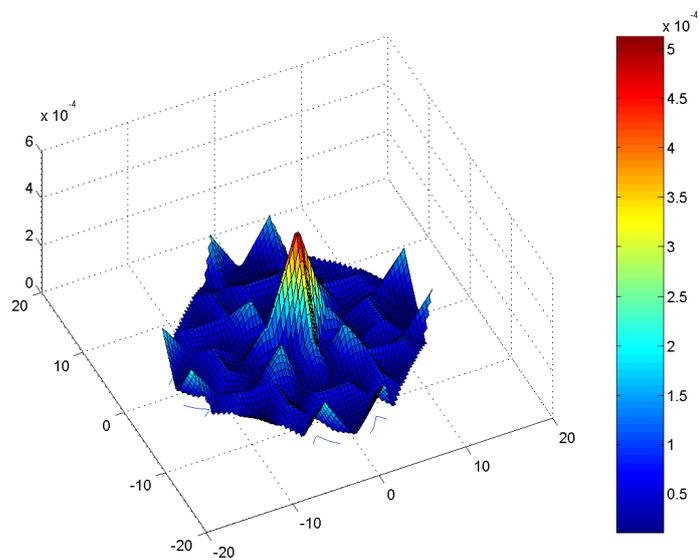
Therefore, for transverse waves [21, 22],  $\omega^2 \rho = C_{44} k^2$  and  $\omega^2 \rho = (C_{11} + C_{12} + 2C_{44}) k^2 / 2$  for [100] and [110] respectively. Table 8.2 lists the main results obtained from the simulation.

### 8.4.2 Thermography

3D thermograms for user-specified plane and time step were calculated using Equation (7.3) on atom positions and velocity detected using the macro method described above. The extracted surfaces were passed onto the MATLAB code, `mesh.m` and plotted. Figure 8.9 is an alternate view of Figure 8.1 showing the plane of the simulated structure for which thermograms were calculated. The impulsed atom is on the opposite face of the structure. Figure 8.8 shows the surface temperature  $T_{av}$ . Figures 8.10–8.11 show the resulting thermograms. The temperature calculation considers the 3D neighborhood of the atom.



(a) surface atom neighborhood



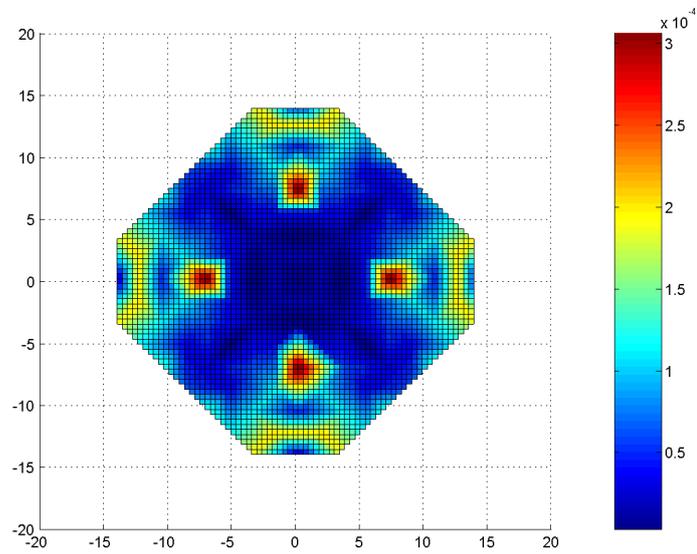
(b) 3D

Fig. 8.9 Surface of structure for which thermographs were calculated and plotted. Figure (b) illustrates a 3D map of the surface showing the surface temperature distribution at termination of simulation.

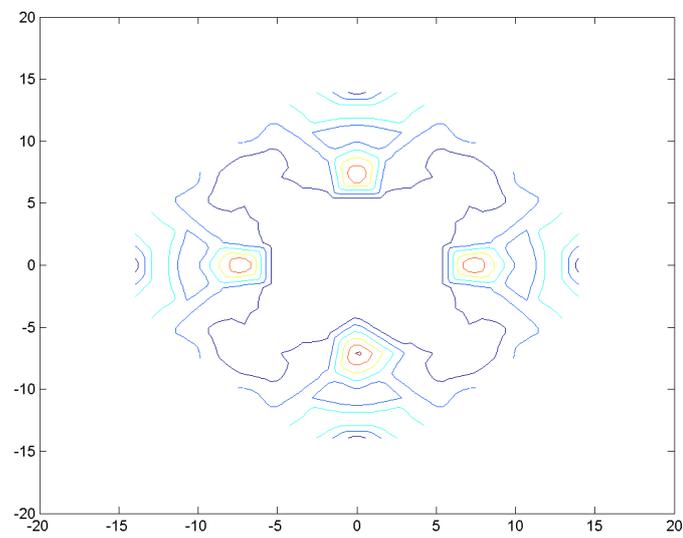
The wave couples vibrational energy to other atoms due to the cascading effect of attractions and repulsions through the lattice. The time lag of a wave between the two faces was used to calculate phase velocity between any two points in the lattice. This gives the wave velocity, which is the speed of sound in copper. Figure 8.6 shows two peaks which correspond to wavelengths of 8.789 Å and 8.152 Å respectively.

## 8.5 Impact

The output of VSV v1.0 is typically a trajectory in the temporary and spatial domain for all particles in the ensemble, as described above. The spatial position of a given particle is available at the integration time step. This leads to a voluminous output which, without visualization, becomes extremely difficult to interpret for the dynamics of the simulated system, and to follow the evolution of the system in phase space. This update therefore provides a visual deciphering of the vast amount of data at the various stages of the simulation thus making deductions about the system easy. The approach taken in writing the update is, as with VSV v1.0, to employ the tools that are readily available to most readers. The code uses transparent VBA scripts in Microsoft Excel to access trajectory data that are loaded into the Excel spreadsheet. The innards of the code are open and heavily commented to enable dissection. An example of an output that is immediately valuable is the .xyz format that can be loaded directly into VMD as simulation frames at the time steps and leisurely visualized. The immediate impacts of this contribution are many fold. First, by facilitating such visualizations, it permits the modelling and following of the behavior of arbitrary coupled phonons from initial impulses in the low-strain elastic model of the fcc lattice. Secondly, by allowing the user to highlight specific atoms in the lattice to which initial impulses are imparted, following the lattice for its responses and propagation of the ensuing bulk and surface waves becomes easier. Thirdly, tracking the wave-based processes that give rise to the macroscopic energetic and thermal properties is easily done for surfaces, lattice

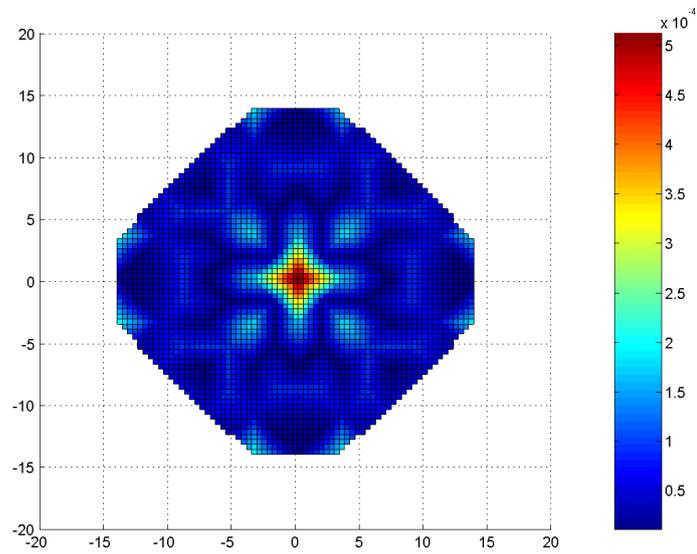


(a) atom neighborhood

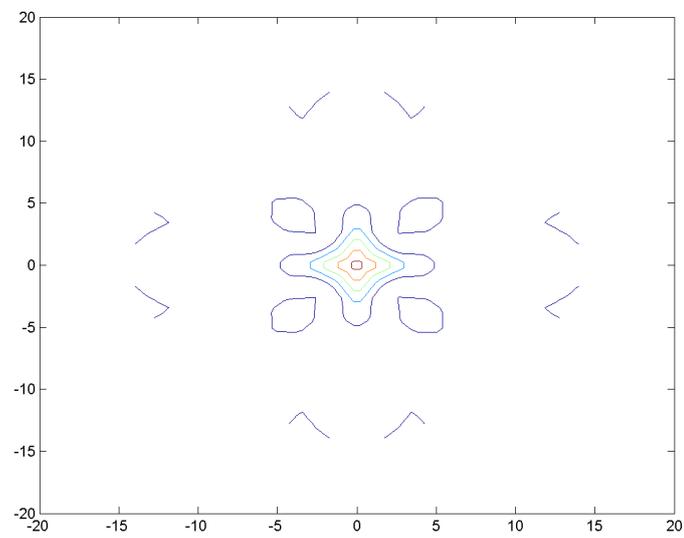


(b) contour

Fig. 8.10 Thermograms of selected lattice surface after 605.55 fs.



(a) atom neighborhood



(b) contour

Fig. 8.11 Thermograms of selected lattice surface after 18854.63 fs.

directions and volumes. Thus, it is possible to derive and visually follow the evolution of the phase space of the system at given instants of time up until the end of the simulation without the necessity of specifying isotherms. In fact, from purely first principles calculations, such visualization has been able to show that steady-state temperature is attained in one given simulated system from purely force-field calculations starting at absolute zero temperature. In other words, without an *a priori* specification of temperature, a perturbed system was able to attain a new steady state temperature above 0 K. Finally, following bond length variations using the software provides a method to correlate system aggregates such as phonon dispersion and the vibrational spectral density. This provides an alternative method to established methods such as the Green-Kubo functionals, and the effective medium theory.

## 8.6 Conclusions

A post-MD simulation processing software has been presented. The article includes an accessible evaluation repository with all sources, that advances the study of nanosized metallic fcc clusters using the Sutton-Chen potential. The collection of source code shows post-processing, how oscillations are detected, how useful surfaces and other parameters are easily extracted and plotted. We suggest a reasonable avenue to determine the vibrational mode frequencies, and how 3D thermograms can be plotted. may be states, calculate elastic parameters and show that the results compare well with the literature. This method arose from the need to process vast amounts of MD output data based on a previously published research for standalone simulations. The method is fully open source. It opens up an important, contemporary area for immediate application by other workers with limited computational resources. The codes were run on a single genuine Intel(R) 3.20 GHz desktop computer having quad core i5-3470 CPU and with 4.0 GB RAM on Windows 7 (64-bit). The simulation of 2281 atoms averaged 12 days on the system. The method was written to complement VSV

---

[7]. The MATLAB codes were developed on version R12. The submission includes a short animation video `Cu - 2281 atoms.mpg` generated from this approach.

# References

- [1] Car R, Parrinello M. Unified approach for molecular dynamics and density functional theory. *Phys. Rev. Lett.*, 55(22):2471–2474., 1985.
- [2] Car R, Parrinello M. The unified approach for molecular dynamics and density functional theory, in simple molecular systems at very high density. *in NATO ASI Series, Series B, Physics, P.P. Loubeyre and N. Boccara (Eds.)*, 186:455–476, 1989.
- [3] Remler DK, Madden PA. Molecular dynamics without effective potentials via the Car-Parrinello approach. *Mol. Phys.*, 70(6):921–66, 1990.
- [4] Tuckerman ME. Ab initio molecular dynamics: basic concepts, current trends and novel applications. *J. Phys.: Condens. Matter.*, 2002. URL [stacks.iop.org/JPhysCM/14/R1297](http://stacks.iop.org/JPhysCM/14/R1297).
- [5] Bintaog G, Xiaorui W, Yujiao C, Zhaohui L, Jianlei Z. High-accuracy infrared simulation model based on establishing the linear relationship between the outputs of different infrared imaging systems. *Infrared Physics & Technology*, 69:155–163, 2015. doi: 10.1016/j.infrared.2015.01.010.
- [6] Ma J, Ma Y, Li C. Infrared and visible image fusion methods and applications: A survey. *Information Fusion*, 45:153–178, 2019. doi: 10.1016/j.inffus.2018.02.004.

- [7] Ocaya R, Terblans JJ. C-language package for standalone embedded atom method molecular dynamics simulations of fcc structures. *SoftwareX*, 5:107–111, 2016. doi: 10.1016/j.softx.2016.05.005.
- [8] Nakano A, Bachlechner ME, Campbell TJ, Kalia RK, Omeltchenko A, Tsuruta K, Vashishta P, Ogata S, Ebbsjo I, Madhukar A. Atomistic simulation of nanostructured materials. *IEEE computational science & engineering*, 5(4):68–78, 1998. doi: 10.1109/99.735897.
- [9] Sharma A, Kalia RK, Nakano A, Vashishta P. Scalable and portable visualization of large atomistic datasets. *Computer Physics Communications*, 163(4):53–64, 2004. doi: 10.1109/99.735897.
- [10] Kittel C. *Introduction to Solid State Physics*. John Wiley & Sons, New Jersey, 2005.
- [11] Pathak LP, Rai RC, Hemkar MP. Lattice vibrations and debye temperatures of transition metals. *Journal of the Physical Society of Japan*, 44(6):1834–1838, 1978.
- [12] Schommers W. in Schommers W, von Blankenhagen P (Eds.) *Structure and Dynamics of Surfaces*. Springer, Berlin, 1986.
- [13] SurfaceDetect.xlsm, FFT.m and mesh.m. Github code repository of demonstration sources. URL <https://github.com/ocayaro/SurfaceDetect>. Updated: 2018-02-10.
- [14] Sutton AP, Chen J. Long-range Finnis-Sinclair potentials. *Philosophical Magazine*, 63(1):139–156, 1990.
- [15] Finnis MW, Sinclair JE. Long-range Finnis-Sinclair potentials. *Philosophical Magazine*, 50:45, 1984.

- [16] Ocaya R, Terblans JJ. Temperature specification in atomistic molecular dynamics and its impact on simulation efficacy. *J. Phys.: Conf. Ser*, 905(012031), 2017. doi: 10.1088/1742-6596/905/1/012031.
- [17] Theoretical and Computational Biophysics Group. Visual Molecular Dynamics. URL [http://www.ks.uiuc.edu/Research/vmd/allversions/what\\_is\\_vmd.html](http://www.ks.uiuc.edu/Research/vmd/allversions/what_is_vmd.html). Accessed: 2017-11-02.
- [18] OpenBabel. XYZ (format). URL <http://openbabel.sourceforge.net/wiki/XYZ>. Assessed: 2018-02-14.
- [19] Ocaya R, Terblans JJ. Coding considerations for standalone molecular dynamics simulations of atomistic structures. *J. Phys.: Conf. Ser*, 905(012018), 2017. doi: 10.1088/1742-6596/905/1/012018.
- [20] Ditlevsen Peter D, Nørmskov Jens K. Vibrational properties of aluminum, nickel and copper surfaces. *Surface Science*, 254:261–274, 1991. doi: 10.1.1.718.7356.
- [21] Chopra KK, Bouarab S. Off-symmetry phonon frequencies of copper. *Physica status solidi (b)*, 125(2):449–504, 1984.
- [22] Jani AR, Gohel VB. On the phonon frequencies of copper in off symmetry directions. *Solid State Communications*, 41(5):407–411, 1982.
- [23] Behari J, Tripathi BB. Frequency spectra and heat capacities of copper, nickel and aluminium. *Aust. J. Phys.*, 23:311–318, 1969. URL <http://adsabs.harvard.edu/full/1970AuJPh..23..311B>.

# Chapter 9

## Conclusions

The preceding chapters detail how we set out initially to devise a standard molecular dynamics platform for a standalone machine to simulate crystalline fcc lattices configurations of up to several thousand atoms. The developed system is intended to include a feature that permits upward scaling of computational power through hardware with a minimal of software reconfiguration. The software is intended for ready use by researchers interested in the atom structures without assuming programming expertise. The motivation is to contribute towards overcoming the high start up inertia that is typical in MD research. In attempting to achieve these overarching goals, specific objectives were identified and followed closely.

The work began by presenting a review of recent, current and growing trends in the field of computation. We then proceeded to weigh the pros and cons of writing our own code versus existing packages and decided on the former approach. The C/C++ language was chosen, and the work to develop a compact library of reusable MD functions. This approach was taken for enhanced performance and also to make it easier to evolve the code to higher versions. The functions themselves implement the Sutton and Chen implementation of the Finnis-Sinclair potential. The modularity of the open library functions makes it easier to modify the code to other crystal systems, such as body-centered cubic (BCC). In the scope of the present work,

BCC and other metallic systems cannot be immediately simulated by the present code, but modifications can readily be done through the library.

The study focus on fcc structures is not particularly limiting since there is an abundance of fcc structures and associated questions that need to be answered. We have stated and computed some answers to such questions using the developed software, called VSV. Such questions involve static and dynamic mechanical or energetic properties. Through these properties the related parameters, such as the equilibrium lattice parameters, temperature, phase space, heat propagation and other properties have been successfully simulated and deduced. The choice of the parameters simulated and evaluated in the present work was guided by the ready availability of published literature data that provided a meaningful basis of comparison between simulation experiment and empirical experiment.

Several considerable challenges were experienced during the work to address as we set out to attain the stated objectives. A necessary, if mandatory aspect of computation that was quickly apparent at the onset of the study was the need to establish parallelization and scalability as an integral aspect of VSV to enhance processing power and reduce execution times. Some evidence of the typical execution time for a simulation was given for the case of 30,261 (+1) atoms, where execution times were successfully reduced from days to a few hours on a single machine. A further design specification was deployability of VSV and its libraries onto a wide range of operating systems. Thus, the operation of VSV was assured on Windows and Linux. The Windows operating system is important since it is the most likely system available to the intended users of VSV. The Linux operating system is considered important from the standpoint of C-coding since it is the native environment for the language and there are routines there that facilitate high performance computing. Apart from the above challenges, issues surrounding the implementation of some of the basic physics were addressed. Early on into the study, the question also arose about how best to simulate temperature in a simulation, which turned out to be a non-trivial question. We have

pointed out in detail the variability in the temperature specifications using two commonly followed thermostats. It became evident that the choice of a simulation thermostat is guided by factors other than merely ease of implementation, and that a clarification was needed. Therefore, we investigated the deterministic, energy-defined thermostat and a stochastic, Monte Carlo (MC) method-based thermostat. In short, ignoring variations and drifts in the system Hamiltonian due to computational error accumulation, the deterministic thermostat was shown to be reversible in time, and canonical in both the Hamiltonian and the potential energy. Consequently, we chose it as being the more descriptive for the class of problems that have been studied in this work. The behavior of the system Hamiltonian can be limited to within acceptable error tolerance bounds of the equilibrium. Also, using the deterministic approach, it is far easier to add external degrees of freedom to simulate environmental system effects such as the surrounding bath temperature, friction and so on. This leads to a more reliable simulation of the dynamical trajectories. However, the method was shown to lack the ability to constrain the total kinetic energy. The MC method is taken to have the main advantage of ease of specifying both the thermostat and the external degrees of freedom.

An interesting observation in the application of VSV was the unwitting detection of a temporal bond length oscillation initiated by perturbation arising from a displaced adatom. VSV follows the propagation of the oscillations. By using a standard mass-spring analogy, we successfully deduced the phonon associated properties of the resultant wave. Thus, a result of this research is of a novel impulse and oscillation method that is useful to calculate the properties for fcc lattices. In addition, this work extends the impulse-oscillation method to postulate a way to initiate wave-like energy transfer through the first Brillouin zone lattice, as constrained by the wave energy. We have also been able to calculate the diffusion properties of the lattice, with emphasis on nanosized lattices. Through standard diffusion models, we suggest a method to calculate diffusion constants in a lattice at low temperature. These calculations suggest that diffusion in such a collection of atoms is due to other factors

other than Brownian motion. The phenomenon of spontaneous atom coalescence, lattice growth and structure formation by atom assimilation were also detected by VSV. These foregoing results were compared with the literature values and the agreements were found to be good. This provides evidence that VSV has achieved its design objectives, and that it has a considerable potential to investigate fcc lattices. The next chapter details some of the published outputs from this research, such as a book chapter, journals and presentations at international conferences, and unpublished manuscripts.

Finally, there is considerable scope for future research in this approach, as summarized in the list below:

- The further development of the VSV code that is contained in the reusable MD function library. The ongoing guidelines continue to be code availability and transparency to other researchers, modularity and evolvability.
- The application of the VSV code to include more fcc systems and their properties through the dynamic adatom concept in this thesis, on different, arbitrary configurations of atoms.
- The improvement of the thermostats and the frequency range of the detected oscillations. The literature suggests that empirical bond-length variations can be as high as 100 GHz. This work has indicated lower frequencies, thus it has focused on low-frequency dynamics and phonons. Investigating higher temperatures implies higher energy phonons.
- The expansion of the concept into a homegrown high performance computing center (HPC) that can be repeated easily will be developed further. This opens up interesting cross-discipline collaboration.

# Chapter 10

## Publications

The following list details some peer-reviewed articles, presentations and software outputs of this study. We also include the manuscripts that are at an advanced state of preparation for peer-review submission.

### 10.1 Refereed journal articles

1. **Ocaya, R.O.**, Terblans, J.J. *C-language package for standalone embedded atom method molecular dynamics simulations of fcc structures*, *SoftwareX* **(5)** 2016, pp. 107-111 doi: 10.1016/j.softx.2016.05.005
2. **Ocaya, R.O.**, Terblans, J.J. *Addressing the challenges of standalone multi-core simulations in molecular dynamics*, *Physical Sciences Reviews* **2(7)** 2016, pp. 13 doi: 10.1515/psr-2016-5100
3. **Ocaya, R.O.**, Terblans, J.J. *Coding considerations for standalone molecular dynamics simulations of atomistic structures*, *Journal of Physics: Conference Series* **905(1)** 2017, pp. 012018 doi: 10.1088/1742-6596/905/1/012018

4. **Ocaya, R.O.**, Terblans, J.J. *Temperature specification in atomistic molecular dynamics and its impact on simulation efficacy*, Journal of Physics: Conference Series **905(1)** 2017, pp. 012031 doi: 10.1088/1742-6596/905/1/012031

## 10.2 Conference presentations

5. **Ocaya, R.O.**, Terblans, J.J. *Coding considerations for standalone molecular dynamics simulations of atomistic structures*, in. Proceedings of the 28th IUPAP Conference on Computational Physics in 2016, Johannesburg, South Africa
6. **Ocaya, R.O.**, Terblans, J.J. *Temperature specification in atomistic molecular dynamics and its impact on simulation efficacy*, in. Proceedings of the 28th IUPAP Conference on Computational Physics in 2016, Johannesburg, South Africa

## 10.3 Chapters in books

7. **R.O. Ocaya**, J.J. Terblans (2017). *Addressing the challenges of standalone multi-core simulations in molecular dynamics*, in. Ponnadurai Ramasami (Editor), Computational Sciences, ISBN: 978-3110465365, September 2017 doi:10.1515/psr-2016-5100

## 10.4 Software

### 10.4.1 Permanent VSV Elsevier code repository

8. **R.O. Ocaya**, VSV code Version 1 - Permanent repository, <https://github.com/ElsevierSoftwareX/SOFTX-D-15-00054>

9. **R.O. Ocaya**, VSV code - Development version, <https://github.com/ocayaro?tab=repositories>

#### 10.4.2 Surface detect applets

10. **R.O. Ocaya**, Surface detection and VSV helper applets, <https://github.com/ocayaro/SurfaceDetect>

### 10.5 Manuscripts under preparation

11. **Ocaya, R.O.**, Terblans, J.J. *Onset of lattice waves and phonons using the Sutton-Chen embedded atom model on nanostructures*, *Wave Motion (ready for submission)*
12. **Ocaya, R.O.**, Terblans, J.J. *VSV 2.0 and VBA integration: a post-simulation visualization approach in low-frequency molecular dynamics*, *Mathematics and Computers in Simulation (ready for submission)*
13. **Ocaya, R.O.**, Terblans, J.J. *Standard molecular dynamics study of spontaneous diffusion and coalescence in nanocluster Cu (100) surfaces near 0K*, *Computational Condensed Matter (in preparation)*
14. **Ocaya, R.O.**, Terblans, J.J. *Deterministic study of structure formation in (100) surfaces of nanosized Cu clusters*, *Advances in Materials Science and Engineering (in preparation)*

# **Appendix A**

## **A guide to using VSV for MD simulations**

### **A.1 Running the VSV code**

The program is written in standard C-language and should therefore be platform independent. In other words, it can be compiled and run in many different environments with little or no modification. This appendix describes how VSV may be run on Microsoft Windows and Ubuntu Linux. It has been tested on the Win32 API-based Windows 7 and Windows 10 operating systems, and several Ubuntu versions.

#### **A.1.1 Windows 7.0 and later**

There are two ways to run the compiled MPI code on a Windows computer, namely through

1. the executable file in the debug folder release,
2. the Integrated Development Environment (IDE) in debug mode.

These ways are discussed separately below.

## A.1.2 The binary executable

The only requirement for this method is to have the executable, the header file, simulation constants and the data file in one folder, say, `fcc`, to be accessed by the executable. Execute the system run command `cmd` as administrator to open the DOS box. Navigate to the folder and type the name of the executable to start the program, e.g.

```
cd \fcc
```

```
velocity-stormer-verlet_02.exe
```

For all purposes this brings the user to the same point of execution described for MS Windows, see **Note A** below.

## A.1.3 On Windows debug

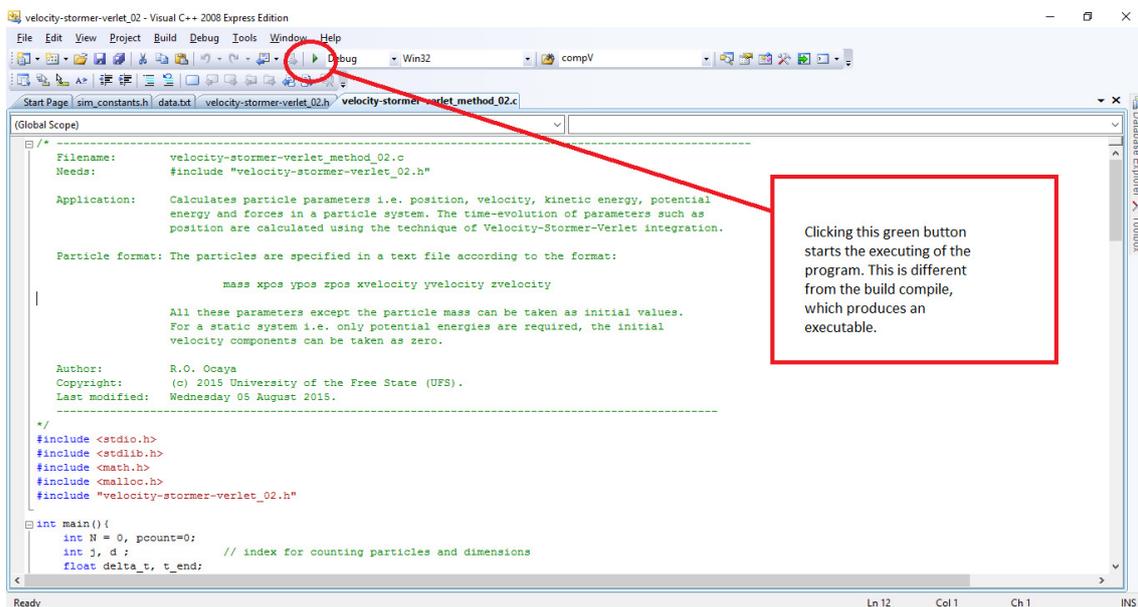


Fig. A.1 The IDE in MS Visual C++ compiler.

The program was developed using Microsoft Visual C++ Express edition 2008. The integrated development environment which has a debug mode to troubleshoot the code. This

environment allows the program and its parameters to be modified and tested instantly before deployment.

**Note A:** After clicking the green button the program execution is at the same stage as the DOS method above. The resulting execution is shown in Figure A.2. In this simple execution, the force and its components on each of the three particles have been calculated are in  $\mu\text{N}$ . The program correctly enumerates the particles as being 3 in total. Examining the components of the force, which for particle  $i$  has the form:

$$F_i = (F_{ix}, F_{iy}, F_{iz}), \quad (\text{A.1})$$

gives the results in Table A.1. Interestingly, the net forces are not zero on each particle but *equal*. This suggests that there is a tendency of each particle to fly apart in the directions determined by the largest force components, although the interplay of the components considered together is such that the arrangement remains in place. This is an illustration of mechanical stability of the array.

Table A.1 Summary of force calculations on the simple 3-atom array in Cu from [1]. All forces are in  $\mu\text{N}$ .

| Atom | $F_{ix}$ | $F_{iy}$ | $F_{iz}$ | Net force ( $F$ ) |
|------|----------|----------|----------|-------------------|
| 1    | 0.0000   | 0.0000   | 0.0001   | 0.0001            |
| 2    | 0.0001   | 0.0000   | 0.0000   | 0.0001            |
| 3    | -0.0001  | 0.0000   | 0.0000   | 0.0001            |

Table A.2 Summary of potential energies in the system of 3-atom array in Cu from [1].

| Energy    | eV        |
|-----------|-----------|
| per atom  | -160.2853 |
| repulsive | 80.1      |
| cohesive  | 240.0     |
| total     | -480.856  |

The calculated potential energies in electron volts (eV) of the individual atoms and of the total system are also shown in Figure A.2. The extent to which these energies are distributed

```

c:\Users\Ocaya\Documents\Visual Studio 2008\Projects\velocity-stormer-verlet_02\Debug\velocity-stormer-verlet_...
This program uses the Sutton-Chen Embedded Atom Method
to calculate potential energies in a particle system.

Found 3 particles and used 312 bytes of total memory.

Particle data from file:

      mass      x      y      z      x_vel      y_vel      z_vel
6.300e+001 -1.000e+000 3.650e-001 0.100e-002 0.000e+000 0.000e+000 0.000e+000
6.300e+001 -3.024e+000 -6.601e-001 -1.521e+000 0.000e+000 0.000e+000 0.000e+000
6.300e+001 -8.469e-001 -3.004e-001 -1.743e+000 0.000e+000 0.000e+000 0.000e+000

E0 = -160.2853
E1 = -160.2853
E2 = -160.2853
Repulsive energy = 6.01e+001 (eV), Cohesive energy = 2.40e+002 (eV).

Total potential energy = -480.8560 (eV)

f_1 = 0.0000
f_2 = 0.0000
Force (uN) on particle 1 = (0.0000, -0.0000, -0.0001), Net force = 0.0001
f_0 = 0.0000
f_2 = 0.0000
Force (uN) on particle 2 = (0.0001, 0.0000, 0.0000), Net force = 0.0001
f_0 = 0.0000
f_1 = 0.0000
Force (uN) on particle 3 = (-0.0001, 0.0000, 0.0000), Net force = 0.0001
Done!
Press a key to exit....

```

Fig. A.2 Example output of the program in Windows during energy calculations.

between repulsion and cohesion are also shown. Table summarizes these energies. The results show that each atom has the same potential energy (-160.3 eV) in the cluster, which is another indication of mechanical stability. The total potential energy of the system is -480.9 eV, which is in exact agreement with the results of Doye et al [1, 2]. The additional information obtained by the VSV code is that the predominant energy in the cluster is cohesive than repulsive, suggesting that the atoms in fact try to condense onto each other rather than fly apart. Interestingly, although copper forms an fcc structure whose conventional unit cell contains 4 atoms and has a coordination number of 8, using only three atoms shows that such a system can exist without any problems.

### A.1.4 Linux e.g. Ubuntu

The program should run on most gcc systems without any problems. The executable can be created as follows (assuming all the above files are included in the same directory):

```
gcc velocity-stormer-verlet_02.c -o VSV_prog -fopenmp -lm ./VSV_prog
```

The first command compiles the code `velocity-stormer-verlet_02.c` using the OpenMP library into an executable called `VSV_prog.o` for the sake of brevity. The second command (alternatively `./VSV_prog.o`) executes the program. The results are identical. The use of a Python script allows a batch of different such programs to be compiled and executed for more complex MD simulations.

## A.2 The GNU General Public License

This code is intended to be used exclusively for research purposes to investigate the interactions of atoms in solid state materials. It is distributed under the terms of GNU General Public License version 3.0. The LICENSE accompanying the code may be viewed at:

<https://github.com/ocayaro/Velocity-Stormer-Verlet/blob/master/LICENSE>

## A.3 VSV code listings

### A.3.1 The library file

The codes that implement the MD functions are grouped together into a single library file called `velocity-stormer-verlet_02.h`, that is registered into a C-application file using the `#include` statement. The library file has functions for force calculation, integration, energy calculations and memory management for a large body of particles.

```
1 // Author: R.O. Ocaya
```

```
2 // (c) 2015, 2016, 2017 University of the Free State (UFS).
3 // First created: 04 April 2015
4 // Last modified: Wednesday 18 January 2017.
5 // Note that printf() is used in program development and
6 // final program output only.
7 //——
8 // define a new type called "Cell".
9 typedef ParticleList* Cell;
10 // function prototypes
11 void timeIntegration_basis (double t, double delta_t, double t_end,
    Particle *p, int N);
12 void outputResults_basis(Particle *p, double N, double t);
13 void computeF_basis(Particle *p, int N);
14 void computeF2_basis(Particle *p, int N);
15 void force(Particle *i, Particle *j);
16 // based on the modified algorithm
17 void force2(Particle *i, Particle *j);
18 void computeX_basis(Particle *p, int N, double delta_t);
19 void computeV_basis(Particle *p, int N, double delta_t);
20 void updateX(Particle *p, double delta_t);
21 void updateV(Particle *p, double delta_t);
22 void compoutStatistic_basis(Particle *p, int N, double t);
23 void particleFileContent(Particle *p, int pcount);
24 void openOutputFile(void);
25 double Utot(Particle *p, int N);
26 double Utot2(Particle *p, int N);
27 void insertList(ParticleList **root_list, ParticleList *i);
28 void deleteList(ParticleList **q);
29 void computeF_LC(Cell *grid, int *nc, float r_cut);
30 void computeX_LC(Cell *grid, int *nc, float *l, float delta_t);
31 void computeV_LC(Cell *grid, int *nc, float *l, float delta_t);
32 void moveParticles_LC(Cell *grid, int *nc, float *l);
```

```

33 double invRootS ( Particle *p, int k, int N);
34 void F_i ( Particle *p, int i, int N);

```

Listing A.1 Function prototypes used in the header file.

```

1 void insertList ( ParticleList **root_list , ParticleList *i){
2     i->next = *root_list;
3     *root_list = i;
4 }
5 void deleteList ( ParticleList **q){
6 // (*q)->next points to element to be removed
7     *q = (*q)->next;
8 }

```

Listing A.2 Particle management for linked-cell method.

```

1 void computeF_LC ( Cell *grid , int *nc , float r_cut){
2     int ic [DIM] , kc [DIM];
3     int d;
4     double r;
5     ParticleList *i , *j;
6     for ( ic [0]=0; ic [0]<nc [0]; ic [0]++)
7     for ( ic [1]=0; ic [1]<nc [1]; ic [1]++)
8     for ( ic [2]=0; ic [2]<nc [2]; ic [2]++)
9     for ( i = grid [index ( ic , nc )]; NULL!=i; i=i->next){
10         for ( d=0; d<DIM; d++)
11             i->p.F[d] = 0;
12         for ( kc [0]=ic [0]-1; kc [0]<=ic [0]+1; kc [0]++)
13         for ( kc [1]=ic [1]-1; kc [1]<=ic [1]+1; kc [1]++)
14             for ( kc [2]=ic [2]-1; kc [2]<=ic [2]+1; kc [2]++){
15 // treat kc [d]<0 and kc [d]>=nc [d] according to boundary conditions;
16 // if (distance of i->p to cell kc <= r_cut)
17             if (1)
18                 for ( j = grid [index ( kc , nc )]; NULL!=j; j=j->next)

```

```

19         if (i!=j){
20             r = 0;
21             for (d=0; d<DIM; d++)
22                 r += sqr(j->p.x[d] - i->p.x[d]);
23             if (r<=sqr(r_cut))
24                 force2(&i->p, &j->p);}}
25 }

```

Listing A.3 Force calculation in linked-cell method.

```

1 void computeX_LC(Cell *grid , int *nc , float *l , float delta_t){
2     int ic [DIM];
3     ParticleList *i;
4     for (ic [0]=0; ic [0]<nc [0]; ic [0]++)
5         for (ic [1]=0; ic [1]<nc [1]; ic [1]++)
6             for (ic [2]=0; ic [2]<nc [2]; ic [2]++)
7                 for (i = grid [index (ic ,nc)]; NULL!=i; i=i->next)
8                     updateX(&i->p, delta_t);
9     moveParticles_LC (grid , nc , l);}
10 void computeV_LC(Cell *grid , int *nc , float *l , float delta_t){
11     int ic [DIM];
12     ParticleList *i;
13     for (ic [0]=0; ic [0]<nc [0]; ic [0]++)
14         for (ic [1]=0; ic [1]<nc [1]; ic [1]++)
15             for (ic [2]=0; ic [2]<nc [2]; ic [2]++)
16                 for (i = grid [index (ic ,nc)]; NULL!=i; i=i->next)
17                     updateV(&i->p, delta_t);
18 }

```

Listing A.4 Position and velocity updater in linked-cell method.

```

1 void moveParticles_LC (Cell *grid , int *nc , float *l){
2     int ic [DIM] , kc [DIM];
3     int d;

```

```

4 ParticleList *i, *q;
5 for (ic[0]=0; ic[0]<nc[0]; ic[0]++)
6   for (ic[1]=0; ic[1]<nc[1]; ic[1]++)
7     for (ic[2]=0; ic[2]<nc[2]; ic[2]++){
8       // pointer to predecessor
9       ParticleList **q = &grid[index(ic,nc)];
10      ParticleList *i = *q;
11      while (NULL != i){
12        // treat boundary conditions for i->x;
13        for (d=0; d<DIM; d++){
14          kc[d] = (int)floor(i->p.x[d] * nc[d] / l[d]);
15          if ((ic[0]!=kc[0]) || (ic[1]!=kc[1]) || (ic[2]!=kc[2])){
16            deleteList(q);
17            insertList(&grid[index(kc,nc)], i);
18          } else q = &i->next;
19          i = *q;}
20    }}

```

Listing A.5 Move particles to their new cells in the linked-cell method.

```

1 void timeIntegration_basis (double t, double delta_t, double t_end,
2   Particle *p, int N){
3   // find the initial force
4   computeF_basis(p, N);
5   // open disk file for output
6   openOutputFile();
7   while (t < t_end){
8     t += delta_t;
9     computeX_basis(p, N, delta_t);
10    computeF_basis(p, N);
11    computeV_basis(p, N, delta_t);
12    compoutStatistic_basis(p, N, t);
13    // print values at each time step

```

```

13     outputResults_basis(p, N, t);}
14     fclose(outputFile);
15 }

```

Listing A.6 Time-based integration function using direct method.

```

1 void timeIntegration2_basis (double t, double delta_t, double t_end,
    Particle *p, int N){
2     // find the initial force
3     computeF2_basis(p, N);
4     // open disk file for output
5     openOutputFile();
6     while (t < t_end){
7         t += delta_t;
8         computeX_basis(p, N, delta_t);
9         // uses the modified algorithm
10        computeF2_basis(p, N);
11        computeV_basis(p, N, delta_t);
12        compoutStatistic_basis(p, N, t);
13        // print values at each time step
14        outputResults_basis(p, N, t);
15    }
16    fclose(outputFile);
17 }

```

Listing A.7 Time-based integration using modified velocity algorithm.

```

1 void openOutputFile(void){
2     if ((outputFile = fopen("outdata.txt", "w")) == NULL){
3         printf("Error: cannot open file for writing...\n");
4         exit (0);}}
5 void outputResults_basis(Particle *p, double N, double t){
6     // int i;

```

```

7  fprintf(outputFile , "%1.3e %1.3e %1.3e %1.3e %1.3e %1.3e %1.3e %1.3e\n
   ", t, p[3].m, p[3].x[0], p[3].x[1], p[3].x[2], p[3].v[0], p[3].v[1],
   p[3].v[2]);
8  }

```

Listing A.8 Disk file output of final results.

```

1  // the naive force computation algorithm , force();
2  void computeF_basis(Particle *p, int N){
3      int i, d, j;
4      for (i=0; i<N; i++)
5          for (d=0; d<DIM; d++)
6              p[i].F[d]=0;
7      for (i=0; i<N; i++)
8          for (j=0; j<N; j++)
9              if (i!=j) force (&p[i],&p[j]);
10 }
11 // the modified force algorithm , force2()
12 void computeF2_basis(Particle *p, int N){
13     int i, d, j;
14     for (i=0; i<N; i++)
15         for (d=0; d<DIM; d++)
16             p[i].F[d]=0;
17     for (i=0; i<N; i++)
18         for (j=i+1; j<N; j++) // note that j = i+1;
19             if (i!=j) force2 (&p[i],&p[j]);
20 }

```

Listing A.9 Functions for the naive and improved force algorithms.

In Listing A.10, the functions `force()` and `force2()` calculate force. The first one uses “naive” approach, where particle interactions  $\bar{F}_{ij}$  and  $\bar{F}_{ji}$  are repeated, which is wasteful of computation time. The second one uses the improved approach, where  $\bar{F}_{ji}$  is accounted for

also during the calculation of  $\bar{F}_{ij}$ , i.e. in the first iteration. Although Listing A.10 relates to the two-body force in Newtonian gravitation, it is included here to illustrate the generality of the method to cover a wide range of different forces. For instance, modifying just this part for forces that are derived from other potentials, such as the  $m$ - $n$  Lennard-Jones potential, allows easy modification of the code to cover different fields.

```

1 void force(Particle *i, Particle *j){
2     int d;
3     double f, r = 0;
4     for (d=0; d<DIM; d++)
5         r += sqr(j->x[d]-i->x[d]);
6     f = (i->m * j->m)/(sqrt(r)*r);
7     for (d=0; d<DIM; d++)
8         i->F[d] += f*(j->x[d]-i->x[d]);
9 }
10 void force2(Particle *i, Particle *j){
11     int d;
12     double f, r = 0;
13     for (d=0; d<DIM; d++)
14         r += sqr(j->x[d]-i->x[d]);
15     // denominator is rij^3
16     f = (i->m * j->m)/(sqrt(r)*r);
17     for (d=0; d<DIM; d++){
18         // this is Fij
19         i->F[d] += f*(j->x[d]-i->x[d]);
20         // this is Fji = -Fij
21         j->F[d] -= f*(j->x[d]-i->x[d]);
22     }}

```

Listing A.10 Functions for the naive and improved force algorithms.

Listing A.11 returns the intermediate sum  $1/\sqrt{S_k}$  used in force calculation in the Sutton-Chen implementation of the Finnis-Sinclair EAM as given by Equation (3.13).

```

1 double invRootS ( Particle *p, int k, int N){
2     int j, d;
3     double r, val = 0;
4     for (j=0; j<N; j++){
5         if (j!=k){
6             r=0;
7             for (d=0; d<DIM; d++){
8                 // r^2 = dx^2 + dy^2 + dz^2
9                 r += sqr(p[j].x[d]-p[k].x[d]);
10                // r = sqrt(r^2)
11                r = pow(r,0.5);
12                // sum (N-1) terms excluding j=k;
13                val += pow(lat_const/r, mint);
14            }
15        }
16        val = 1/pow(val,0.5);
17        // return reciprocal of square root
18        return (val);
19    }

```

Listing A.11 Calculation of the  $S_k$  intermediate term of the SC potential.

Listing A.12 defines a function that receives the complete particle array, computes and returns the force on the  $i$ -th particle using the Sutton-Chen potential.

```

1 void F_i ( Particle *p, int i, int N){
2     int j, d;
3     double r, r2, rn, f;
4     double iterm, jterm;
5     for (j=0; j<N; j++){
6         if (j!=i){
7             r=0;
8             // determine rij for each i and j pair

```

```

9     for (d=0; d<DIM; d++)
10         // r^2 = dx^2 + dy^2 + dz^2
11         r += sqr(p[j].x[d]-p[i].x[d]);
12     // r2 = r^2
13     r2 = r;
14     // rij = sqrt(r^2)
15     r = pow(r,0.5);
16     // rn = (sigma/rij) in Griebel
17     rn = lat_const/r;
18     iterm = invRootS(p,i,N);
19     jterm = invRootS(p,j,N);
20     f = -eps*(nint*pow(rn,nint)-0.5*cn*mint*(iterm + jterm)*pow(rn,
mint))/r2;
21     // printf("\nf_%d = %.4f\n", j, f); // debugging only
22     // for each i, update sum up forces Fij for each dimension
23     // and store the force components
24     for (d=0; d<DIM; d++)
25         p[i].F[d] += f * (p[j].x[d] - p[i].x[d]);
26     }}
27 }

```

Listing A.12 Force  $F_{ij}$  calculation in Sutton-Chen method.

Listing A.13 computes the Lennard-Jones 12-6 pairwise particle force.

```

1 void force_LJ(Particle *i, Particle *j){
2     double r, s, f = 0;
3     int d;
4     for (d=0; d<DIM; d++)
5         r += sqr(j->x[d] - i->x[d]);
6     // here r=sqr(rij)
7     s = sqr(lat_const) / r;
8     s = sqr(s) * s;
9     // here s=pow(sigma/rij,6)

```

```

10  f = 24 * eps * s / r * (1 - 2 * s);
11  for (d=0; d<DIM; d++)
12      i->F[d] += f * (j->x[d] - i->x[d]);
13  // store force components
14  }

```

Listing A.13 Lennard-Jones 12-6 force  $\bar{F}_{ij}$  calculation.

Listing A.14 shows the functions that update the instantaneous positions and velocities of the particles relative to the entire particle ensemble.

```

1 void computeX_basis(Particle *p, int N, double delta_t){
2     int i;
3     for (i=0; i<N; i++)
4         updateX(&p[i], delta_t);
5 }
6 void computeV_basis(Particle *p, int N, double delta_t){
7     int i;
8     for (i=0; i<N; i++)
9         updateV(&p[i], delta_t);
10 }
11 void updateX(Particle *p, double delta_t){
12     int d;
13     double a = delta_t * .5 / p->m;
14     for (d=0; d<DIM; d++){
15         p->x[d] += delta_t*(p->v[d] + a * p->F[d]);
16         p->F_old[d] = p->F[d];}
17 }
18 void updateV(Particle *p, double delta_t){
19     int d;
20     double a = delta_t * .5 / p->m;
21     for (d=0; d<DIM; d++)
22         p->v[d] += a * (p->F[d] + p->F_old[d]);

```

23 }

Listing A.14 Updater functions in the Lennard-Jones 12-6 force  $\bar{F}_{ij}$  calculation.

Listing A.15 computes the total kinetic energy  $E$  of the particle ensemble.

```

1 // print kinetic energy e at time t
2 void compoutStatistic_basis (Particle *p, int N, double t){
3     double v, e = 0;
4     int i, d;
5
6     for (i=0; i<N; i++){
7         v = 0;
8         for (d=0; d<DIM; d++)
9             v += sqr(p[i].v[d]);
10        e += .5 * p[i].m * v;}
11 }

```

Listing A.15 Total ensemble kinetic energy calculation.

Listing A.16 displays the contents of the particle data output file created by the user. This function is useful to verify that the particle list is present and correct before the simulation.

```

1 void particleFileContent (Particle *p, int pcount){
2     int i;
3     printf("\nParticle data from file:\n");
4     printf("\n%10s %10s %10s %10s %10s %10s %10s\n", str1, str2, str3,
5         str4, str5, str6, str7);
6     for (i=0; i<pcount; i++)
7         printf("%1.3e %1.3e %1.3e %1.3e %1.3e %1.3e %1.3e\n", p[i].m, \
8         p[i].x[0], p[i].x[1], p[i].x[2], p[i].v[0], p[i].v[1], p[i].v[2]);

```

Listing A.16 Verification of particle contents before simulation.

Listing A.17 computes the total potential energy,  $U_{tot}$ , in the entire particle ensemble.

```
1 double Utot(Particle *p, int N){
2     double rho_i=0, V_i=0, a_sum=0, b_sum=0, U_sum=0;
3     double r=0;
4     int i, j, d;
5     for (i=0; i<N; i++){
6         // outer summation of rho for all N
7         a_sum = 0;
8         // reset inner_sum for each i value
9         for (j=0; j<N; j++){
10            if (j!=i){
11                r=0;
12                for (d=0; d<DIM; d++){
13                    // find rij
14                    r += sqrt(p[j].x[d]-p[i].x[d]);
15                    r = pow(r,0.5);
16                    // printf("\nr%d_%d=%0.4f", i, j, r);
17                    // interatomic distances, for debugging only
18                    // note: printf() consumes the most simulation time if included
19                    // sum for (N-1) terms ie exclude j=i;
20                    a_sum += pow(lat_const/r, nint);}
21            }
22            a_sum = 0.5*a_sum;
23            // reset inner sum for each i value
24            b_sum = 0;
25            for (j=0; j<N; j++){
26                if (j!=i){
27                    r=0;
28                    for (d=0; d<DIM; d++){
29                        r += sqrt(p[j].x[d]-p[i].x[d]);
30                        // r^2 = dx^2 + dy^2 + dz^2
31                        r = pow(r,0.5);
32                        // r = sqrt(r^2)
```

```

33     // printf("\nr%d_%d=%.4f", i, j, r);
34     // interatomic distances , for debugging only
35     b_sum += pow(lat_const/r, mint);
36     // sum for (N-1) terms ie exclude j=i;
37     }
38 }
39 b_sum = cn*sqrt(b_sum);
40 // find sqrt of inner sum
41 printf("\nE%d = %.4f", i, a_sum-b_sum);
42 // for debugging only
43 U_sum += (a_sum - b_sum);
44 // update U_sum
45 }
46 // for debugging only
47 printf("\nRepulsive energy = %.2e (eV), Cohesive energy = %.2e (eV).",
48     a_sum, b_sum);
49 return (eps*U_sum);
}

```

Listing A.17 Total ensemble potential energy calculation.

### A.3.2 The main program

Listing A.18 is the main application for any MD simulation using the functions within the library header file. It calculates particle parameters i.e. position, velocity, kinetic energy, potential energy and forces in the arbitrary particle ensemble system. The time-evolution of the particle parameters are calculated using the Velocity-Stormer-Verlet integration.

```

1 #include <stdio .h>
2 #include <stdlib .h>
3 #include <math .h>
4 #include <malloc .h>

```

```
5 #include "velocity-stormer-verlet_02.h"
6 int main() {
7     int N = 0, pcount=0;
8     // index for counting particles and dimensions
9     int j, d ;
10    float delta_t, t_end;
11    // initial values from file
12    float mass_0=0, x_0=0, y_0=0, z_0=0, v_x0=0, v_y0=0, v_z0=0;
13    double total_pot_energy=0, ParticleForce = 0, netForce = 0;
14    Particle *p;
15    FILE *particleFile;
16    // find #particles in system and
17    // allocate memory dynamically
18    if ((particleFile = fopen("data.txt", "r")) == NULL){
19        printf("Error: cannot open file for reading...\n");
20        exit (0);
21    }
22    else {
23        while (fscanf(particleFile, "%f %f %f %f %f %f %f", &mass_0, &x_0, &
24            y_0, &z_0, &v_x0, &v_y0, &v_z0) != EOF)
25            pcount++;
26        N = pcount;
27    }
28    // reset file read ptr to start of file
29    rewind(particleFile);
30    pcount = 0;
31    (Particle *)p = (Particle *)malloc(N*sizeof(*p));
32    // User greeting
33    printf("This program uses the Sutton-Chen Embedded Atom Method \n");
34    printf("to calculate potential energies in a particle system.\n\n");
35    // Memory usage statistics
```

```

35  printf("Found %d particles and used %d bytes of total memory.\n", N, N
    * sizeof(*p));
36  // populate memory array with particle data
37  while(fscanf(particleFile, "%f %f %f %f %f %f", &mass_0, &x_0, &y_0
    , &z_0, &v_x0, &v_y0, &v_z0) != EOF){
38      p[pcount].m = mass_0;
39      p[pcount].x[0] = lat_const*x_0;
40      p[pcount].x[1] = lat_const*y_0;
41      p[pcount].x[2] = lat_const*z_0;
42      p[pcount].v[0] = v_x0;
43      p[pcount].v[1] = v_y0;
44      p[pcount].v[2] = v_z0;
45      // reset all forces
46      p[pcount].F_old[0] = 0;
47      p[pcount].F_old[1] = 0;
48      p[pcount].F_old[2] = 0;
49      p[pcount].F[0] = 0;
50      p[pcount].F[1] = 0;
51      p[pcount].F[2] = 0;
52      pcount++;
53  }
54  particleFileContent(p, pcount);
55  // use timeIntegration_basis() or timeIntegration2_basis()
56  // e.g. timeIntegration2_basis(0, delta_t, t_end, p, N);
57  // Calculate Utot
58  total_pot_energy = Utot(p, pcount);
59  printf("\n\nTotal potential energy = %.4f (eV)\n\n", total_pot_energy)
    ;
60  for (j=0; j<N; j++){
61      F_i(p, j, pcount);
62      // net force on particle j
63      netForce = 0;

```

```
64     for (d=0; d<DIM; d++)
65         // vector sum of force components
66         netForce += pow(p[j].F[d], 2);
67     // in micro Newtons
68     netForce = pow(netForce,0.5);
69     printf("\nForce (uN) on particle %d = (%.4e, %.4e, %.4e), Net force
70         = %.4e", j+1, p[j].F[0], p[j].F[1], p[j].F[2], netForce);
71 }
72 printf("\nDone!\n\nPress a key to exit....");
73 getch();
74 // free up all allocated memory
75 free(p);
76 fclose(particleFile);
77 return (0);
78 }
```

Listing A.18 Listing of a typical main program showing output of Utot.

## A.4 Helper applications

### A.4.1 Conversion to VMD format by Excel macro

The VSV main program generates vast amounts of trajectory output points of the format  $(t, \bar{r}, \bar{v})$  for the complete ensemble at each time-step  $t$ . Thus, for all the available degrees of freedom e.g.  $N$  lattice particles sampled over  $M$  simulation time steps, there are  $(7 \times N \times M)$  individual double precision numbers in the output. Even for a small array of particles comprising only 373 atoms, with a total 2000 total steps then there are at least 5 million individual such numbers in the output trajectory, corresponding to at least 96 megabytes of data. This necessitates output visualization if meaningful deductions are to be made from the simulation.

In order to visualize the output, the data was loaded into Microsoft Excel and processed by a purposely written Visual Basic macro into a standard trajectory ".xyz" file. This file was then loaded into the Visual Molecular Dynamics (VMD) program developed by the Molecular Dynamics Group at University of Illinois at Urbana–Champaign and made freely available [3]. VMD has several important features beside visualization. For instance, specific atoms and bonds can be selected and their behavior over the simulation time can be saved as secondary numerical output. This is the method that was used to generate the data in Chapter 6, which allowed the impact generated oscillations to be quantified. This led to a new approach, reported in this research for the first time to highlight bond oscillations within the context of impact generated phonons. Chapter 8 shows, for the first time, how spectral responses to phonons can be related to particle displacements, thereby solving a persistent problem in MD thus far. Figure A.3(a) shows only a few lines at the beginning and end of an actual output of VSV that are operated on by the macro in Figure A.3(b) to generate the secondary output file for visualization in VMD.

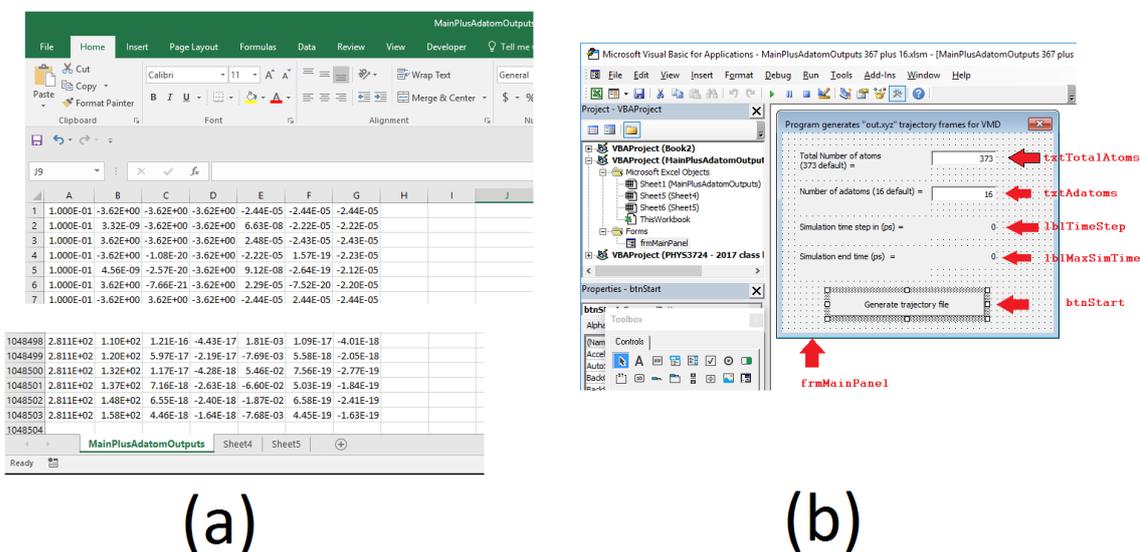


Fig. A.3 Screen shots showing (a) a few entries in the raw output of VSV, and (b) interface design for interaction between macro and VSV output data in generation of secondary output for visualization in VMD. This particular simulation has 2811 simulation time instants. The VB form objects labeled in red refer to the handles in Listing A.19.

An additional, useful feature of VMD is its ability to visualize one frame of output at a time, representing each simulation time step. It also creates good quality AVI format video files for the complete simulation. Listing A.19 shows the Visual Basic code within Microsoft Excel that processes the VSV output into usable data for VMD.

```
1 ' this program creates a single file with multiple .xyz type data
2 ' to enable easy visualization in VMD. Each .xyz type data becomes
3 ' a frame when loaded into VMD. having one file allows a
4 ' single file load into VMD which dramatically saves time in
5 ' creating frames.
6 '
7 ' (c) Ocaya, R.O.
8 ' 17 June 2017
9 '
10 Private Sub btnStart_Click ()
11     Dim i, j, k, TotalAtoms, Adatoms As Integer
12     Dim t, x, y, z, v1, v2, v3 As Double
13     Dim TimeStep,RowIndex As Long
14     Dim oldname, filename As String
15     Dim atomtype As String
16
17     TotalAtoms = Val(txtTotalAtoms.Text)
18
19     ' user specifies number of adatoms
20     Adatoms = Val(txtAdatoms.Text)
21
22     ' find simulation time step from worksheet data
23
24     TimeStep = Sheet1.Cells(1 + TotalAtoms, 1) - Sheet1.Cells(1, 1)
25     ' report the calculated value
26     lblTimeStep.Caption = Str(TimeStep)
27
```

```
28 ' count non-blank rows of the worksheet data. The last row index
29 ' will give us the maximum simulation time, hence the number of
30 ' rows in the data. This is then used to group data in the XYZ file
31 i = 1
32 While Sheet1.Cells(i, 1) <> ""
33     i = i + 1
34 Wend
35 RowIndex = i - 2
36
37 ' report the simulation time found in the worksheet
38 lblMaxSimTime.Caption = Str(Sheet1.Cells(RowIndex, 1))
39
40 atomtypeCu = "Cu" ' we are interested in copper
41 atomtypeNi = "Ni" ' use this as a trick to highlight Cu as Ni
42 filename = "out.xyz"
43
44 Open filename For Output As #1
45 j = 0
46 k = 1
47 For i = 0 To RowIndex ' default RowIndex = 37299
48     If (i Mod TotalAtoms = 0) Then
49 ' count the number of
50 ' particles in the file
51     ' Close #1
52     j = j + 10
53     k = 1
54     ' this is the number of atoms in xyz file
55     Print #1, TotalAtoms
56     Print #1, "Created by Ocaya"
57 End If
58 t = Sheet1.Cells(i + 1, 1)
59 x = Sheet1.Cells(i + 1, 2)
```

```

60  y = Sheet1.Cells(i + 1, 3)
61  z = Sheet1.Cells(i + 1, 4)
62  v1 = Sheet1.Cells(i + 1, 5)
63  v2 = Sheet1.Cells(i + 1, 6)
64  v3 = Sheet1.Cells(i + 1, 7)
65  ' Write #1, j, x, y, z, v1, v2, v3
66  ' Write #1, Str(atomtype) + " " + Str(x) + " " + Str(y) + " " + Str(z)
67  ' default = 357 i.e. (total atoms - adatoms)=357
68  If k <= (TotalAtoms - Adatoms) Then
69  ' suppress commas by using Print rather than Write
70  Print #1, atomtypeCu, x, y, z
71  Else
72  Print #1, atomtypeNi, x, y, z
73  End If
74  k = k + 1
75  Next i
76  Close #1
77  Exit Sub
78 End Sub
79
80 Private Sub UserForm_Initialize()
81  txtAdatoms.Text = 16
82  txtTotalAtoms.Text = 373
83 End Sub

```

Listing A.19 Code converts VSV output to VMD output for visualization.

## A.4.2 Spectral response calculation

Listing A.20 calculates the Fast Fourier Transform (FFT) of bond length oscillations to give an arbitrary count versus frequency of all detected phases. It was vital in linking the spectral response of phonons in the lattice to lattice atom displacements for the first time.

```
1 %% myFFT.m
2 %%
3 %% This m.file loads a trajectory file from the bond-length
4 %% versus time simulation, calculates the fast fourier transform
5 %% of the amplitude i.e the phase/frequency behavior, and then
6 %% writes the output into the file 'fftout.txt'. The data in this
7 %% file can then be plotted in another environment, such as Origin
8 %%
9 %% Requirement: the trajectory data is saved in real-time/bond
10 %% length amplitude in the file
11 %%           'fft_in.txt'
12 %%
13 %% (c) Ocaya – November 2017
14 %%
15 % Assuming data is in two columns, time and amplitude
16 load -ascii fft_in.txt x y;
17 n=length(fft_in);
18 t=fft_in(1:n,1);
19 y=fft_in(1:n,2);
20
21 %% This is the approximate average location. I am doing this to
22 %% increase the height of the FFT plots (i.e. signal-noise ratio)
23 %% any other would not change the frequency, but only decrease
24 %% the height. This is to make the plots easier to see.
25
26 % Show the raw amplitude vs time data
27 % But first calculate the average value or DC level
28 av=sum(y)/length(y);
29 yy=y-av;
30 %% yy is ordinarily the "dc level" of the signal i.e. average value
31 plot(t,y)
32 dt=t(2)-t(1);
```

```
33 N=length(t);
34 fs = 1/dt; % sampling frequency
35 df=fs/(N-1);
36 f=-fs/2:df:fs/2;
37 % compute the FFT transform of the signal amplitudes
38 ytran = abs(fft(yy));
39 plot(f,ytran) % plot the phases versus frequency
40 % put in single column format to save into ASCII file
41 f= transpose(f);
42 % create an array of similar dimensions to create
43 %% output text file of (f, fft)
44 fft_out = fft_in;
45 % populate it with frequency data in column 1
46 fft_out(1:1:n,1) = f;
47 % and output FFT data in column 2
48 fft_out(1:1:n,2) = ytran;
49 save -ascii -double -tabs fftout.txt fft_out;
```

Listing A.20 Code calculates the spectral response through FFT of bond length variation with time.

# References

- [1] Doye JPK, Wales DJ. Global minima for transition metal clusters described by the Sutton-Chen potentials. *New J. Chem.*, pages 733–744, 1998.
- [2] Wales DJ, Doye JPK. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. *J. Phys. Chem. A.*, 101 (28):5111–5116, 1998.
- [3] Theoretical and Computational Biophysics Group. Visual Molecular Dynamics, 2017. URL [http://www.ks.uiuc.edu/Research/vmd/allversions/what\\_is\\_vmd.html](http://www.ks.uiuc.edu/Research/vmd/allversions/what_is_vmd.html). Accessed: 2017-11-02.