PREDICTING THE THERMODYNAMIC PROPERTIES OF GOLD NANOPARTICLES USING DIFFERENT FORCE FIELDS

A Dissertation

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

Yongjin Park

December, 2010
PREDICTING THE THERMODYNAMIC PROPERTIES OF GOLD
NANOPARTICLES USING DIFFERENT FORCE FIELDS

Yongjin Park
Dissertation

Approved: 

Advisor 
Dr. Gustavo A. Carri

department Chair 
Dr. Ali Dhinojwala

Committee Member 
Dr. Mark D. foster

Committee Member 
Dr. Ali Dhinojwala

Committee Member 
Dr. Gary R. Hamed

Committee Member 
Dr. Jutta Luettmer-Strathmann

Accepted:

Advisor 
Dr. Gustavo A. Carri

Department Chair 
Dr. Ali Dhinojwala

Dean of the College 
Dr. Stephen Z. D. Cheng

Dean of the Graduate School 
Dr. George R. Newkome

Date 

Date

Date
ABSTRACT

The objective of this research was to learn how to predict the thermodynamic properties of gold nanoparticles using computational tools. The lowest energy structures of gold nanoparticles of various sizes were determined and thermodynamic properties such as the free energy ($F$), internal energy ($U$), entropy ($S$), and specific heat ($C_v$) of the gold nanoparticles were investigated using a fully-atomistic Monte Carlo simulation method that utilizes a modified Wang-Landau algorithm. Eight well-known force fields for metallic systems were employed to model gold nanoparticles: the Lennard-Jones potential (LJ), the Lennard-Jones potential with Heinz’s parameterization (LJH), the Gupta potential, the Sutton-Chen potential (SC), the Sutton-Chen potential with Pawluk’s parameterization for small clusters (SCP), the Quantum Sutton-Chen potential (Q-SC), the Embedded Atom Method (EAM) by Cai and Ye, and the empirical potential for gold proposed by Olivier and coworkers (POT). Subsequently, we explored the accuracy of each force field in the description of the thermodynamic behavior of gold nanoparticles.

The thermodynamic properties of gold nanoparticles were computed from the Density of States which was obtained as a result of the Monte Carlo simulation. Afterwards, the melting point of gold nanoparticles was determined from the behavior of the calculated thermodynamic properties and was compared with theory, experimental
observations and other simulation results. The force fields employed predicted melting points of gold nanoparticles over a wide range of temperatures. A thorough comparison with the available experimental observations showed that the Quantum Sutton-Chen potential (Q-SC) correctly described the melting behavior of gold nanoparticles with sizes smaller than 1.3 nanometers.
ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Gustavo A. Carri for his continuous support and encouragement throughout the period of study as a graduate student. I would not be able to write this dissertation without his excellent guidance. Dr. Carri has always been an inspiration for me and helped me keep in track during the course of my studies at The University of Akron.

I would also like to thank my committee members: Dr. Mark D. Foster, Dr. Ali Dhinojwala, Dr. Gary R. Hamed, and Dr. Jutta Luettmer-Strathmann for their insightful comments and valuable discussions and suggestions.

I would also like to thank my group members: Dr. Pavan K. Ghatty (currently at Oak Ridge National Laboratory), Dr. Andrew Nathan (currently at Bryan Research & Engineering, Inc.), Xin Li, and Jing Yu for the numerous discussions we had and for being such good friends of mine.

I would also like thank my friends: Dr. Kwang Un Jeong, Dr. Suk Hoon Bum, Dr. Byung Jo Lee, Dr. Namil Kim, Eun Kyung Kim, Kwang Su Seo, Jae Sun Choi, Jin Kuk Lee, Hyungjin Lee, Abhishek Banerjee, Dr. Carlos Barrios, Vijay Chavan, Kurt Chiang, Andrew Heidenreich, Dr. Someswara Peri, and Chien-Lung Wang who helped make my life in Akron more enjoyable. Dr. Yun Ho Kim, Hyun Ok Ham, Dr. Sung Kwang Lim, Dr.
Sun Jung Yoon, Jin Sung Lee, Kyu Hak Oh, Sang Hee Lee, and my best friend Suk Hwan Hong were always there when I needed them.

Last but not least, I would like to show my best appreciation to my parents Jae Myung Park and Sun Suk Hong, to my wife Jee Yeon Yeom, to my parents in law Young Chul Yeom and Jin Won Seo, to my brother Seongjin Park, and to my lovely daughter Olivia Yein Park. They have supported me financially and emotionally throughout my entire life. I would not be able to finish this dissertation without their bottomless love and encouragement.
# TABLE OF CONTENTS

| LIST OF TABLES | ix |
| LIST OF FIGURES | xi |

## CHAPTER

### I. METALLIC NANOPARTICLES .................................................................1

1.1 Nanoparticles ......................................................................................2

1.2 Metallic nanoparticles ........................................................................3

1.3 Structural properties of metallic nanoparticles ..............................7

1.4 Thermodynamic properties of metallic nanoparticles .................17

1.5 Force Fields for metallic systems ................................................20

1.6 Motivation of the study ...................................................................34

### II. INTRODUCTION TO MONTE CARLO SIMULATION ....................37

2.1 Fundamentals of Monte Carlo simulation ......................................37

2.2 Simple Sampling .............................................................................38

2.3 Importance Sampling (Metropolis algorithm) ..................................38

2.4 Wang-Landau algorithm and the Density of States .......................41

2.5 Wang-Landau algorithm with global updates ...............................44

### III. SIMULATION PROTOCOL ...............................................................46

3.1 The modified Wang-Landau algorithm ........................................46
3.2 Force Fields for gold nanoparticles .................................................................48
3.3 Simulation methodology .............................................................................54

IV. THERMODYNAMIC PROPERTIES OF GOLD NANOPARTICLES ............59
  4.1 Estimation of the Density of States ............................................................59
  4.2 Calculation of the thermodynamic properties .........................................60
  4.3 Melting of gold nanoparticles .................................................................80
  4.4 Conclusions ............................................................................................95

V. CONCLUSIONS .............................................................................................96

BIBLIOGRAPHY ...............................................................................................98

APPENDIX ........................................................................................................105
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Non-icosahedral global minima of Lennard-Jones clusters. N: number of atoms, TO: truncated-octahedral, mDh: Mark-decahedral, Td: tetrahedral [35-54]</td>
</tr>
<tr>
<td>1.2</td>
<td>The global minima of gold (Au), silver (Ag), and gold-silver (Au-Ag) alloys. Ih: icosahedral, Oh: octahedral, TO: truncated-octahedral, mDh: Mark-decahedral, TC: truncated-cube, CO: cubeoctahedral. Oh, TO, cube, TC, and CO belong to the face-centered cubic (fcc) structure [39, 56, 57]</td>
</tr>
<tr>
<td>1.3</td>
<td>Parameters of the RGL potential [91] and of the Gupta potential [90] for face-centered cubic (fcc) metals</td>
</tr>
<tr>
<td>1.4</td>
<td>Parameters of the Sutton-Chen potential (SC) and the Quantum Sutton-Chen potential (Q-SC) for several face-centered cubic (fcc) metals [92, 95]. ( \epsilon ) = meV, ( a ) = Å, and c, m, and n are dimensionless</td>
</tr>
<tr>
<td>3.1</td>
<td>Parameters and coefficients for the POT potential. ( R_c, R_d, R_e, ) and ( R_M ) are in Å and the others are dimensionless [92]</td>
</tr>
<tr>
<td>3.2</td>
<td>Size of the atomic clusters studied with each force field</td>
</tr>
<tr>
<td>3.3</td>
<td>The diameter of gold nanoparticles (twice the radius of gyration) predicted by various force fields. Units are nanometers</td>
</tr>
<tr>
<td>4.1</td>
<td>Melting point of gold nanoparticles as predicted by the various force fields. N is the number of atoms and D is the diameter. * represents the clusters whose melting point could not be identified. The melting points are in Kelvin</td>
</tr>
</tbody>
</table>
4.2 Standard deviation of the predicted melting points from the least square fit of them with the experimental data  

4.3 The slope of the linear regression fit of the predicted melting points and of the least square fit at the intersection with the linear regression fit. For the Quantum Sutton-Chen potential (Q-SC), the slope of the least square fit was obtained at the median of the studied sizes.
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Rayleigh light-scattering of gold (Au) and silver (Ag) nanoparticles displaying various colors depending on the material, shape, and size. (Reprinted with permission from [9]. Copyright 2001 American Association for the Advancement of Science)</td>
</tr>
<tr>
<td>1.2</td>
<td>Schematic illustration of Surface Plasmon Resonance (SPR) - Interaction between light and confined electrons of a metal nanoparticle</td>
</tr>
<tr>
<td>1.3</td>
<td>Crystallographic structures of metallic nanoparticles</td>
</tr>
<tr>
<td>1.4</td>
<td>Illustration of Mackay- and anti-Mackay- atom packing</td>
</tr>
<tr>
<td>1.5</td>
<td>Melting point of gold nanoparticle as a function of its size. (○) experimental [69], (△) experimental [75], (■) MD simulation with MEAM [71], and (▼) MD simulation with EAM [72] (Reprinted with permission from [71]. Copyright 2002 Elsevier)</td>
</tr>
<tr>
<td>2.1</td>
<td>Schematic picture of the energy landscape illustrating the concept of “Importance Sampling”</td>
</tr>
<tr>
<td>3.1</td>
<td>Density of States (DOS) of 55-atom Lennard-Jones cluster estimated with different values of final modification factor (f)</td>
</tr>
<tr>
<td>3.2</td>
<td>Calculated heat capacity (Cv) for the 55-atom Lennard-Jones cluster estimated with different values of the final modification factor (f)</td>
</tr>
<tr>
<td>3.3</td>
<td>Schematic illustration of a “Parallel Simulation”</td>
</tr>
</tbody>
</table>
4.1 Logarithm of the Density of States $g(E)$ of gold nanoparticles described by the Lennard-Jones potential. $\varepsilon$ is the Lennard-Jones parameter .................................................................60

4.2 Specific heat ($C_v$) of Lennard-Jones clusters as a function of temperature. The upper plot shows the results of this study while the lower figures show the results of other Monte Carlo simulation [59]. The numbers on the y-axis represent the number of atoms in the clusters (Reprinted with permission from [59]. Copyright 2006 American Institute of Physics) .............................................63

4.3 Calculated free energy ($F$) of gold nanoparticles described by the Lennard-Jones potential (LJ) as a function of temperature ........................................................................................................64

4.4 Calculated internal energy ($U$) of gold nanoparticles described by the Lennard-Jones potential (LJ) as a function of temperature ........................................................................................................64

4.5 Calculated entropy ($S$) of gold nanoparticles described by the Lennard-Jones potential (LJ) as a function of temperature ........................................................................................................65

4.6 Calculated specific heat ($C_v$) of gold nanoparticles described by the Lennard-Jones potential (LJ) as a function of temperature ........................................................................................................65

4.7 Specific heat ($C_v$) of gold nanoparticles described by the Lennard-Jones potentials. Solid line: original parameterization, ($\circ$): Heinz’s parameterization .............................................67

4.8 Calculated free energy ($F$) of gold nanoparticles described by the Gupta potential as a function of temperature ........................................................................................................68

4.9 Calculated internal energy ($U$) of gold nanoparticles described by the Gupta potential as a function of temperature ........................................................................................................68

4.10 Calculated entropy ($S$) of gold nanoparticles described by the Gupta potential as a function of temperature ........................................................................................................69

4.11 Calculated specific heat ($C_v$) of gold nanoparticles described by the Gupta potential as a function of temperature ........................................................................................................69
4.12 Calculated free energy \( (F) \) of gold nanoparticles described by the Sutton-Chen potential (SC) as a function of temperature .................................................................70

4.13 Calculated internal energy \( (U) \) of gold nanoparticles described by the Sutton-Chen potential (SC) as a function of temperature .................................................................70

4.14 Calculated entropy \( (S) \) of gold nanoparticles described by the Sutton-Chen potential (SC) as a function of temperature .................................................................71

4.15 Calculated specific heat \( (C_v) \) of gold nanoparticles described by the Sutton-Chen potential (SC) as a function of temperature .................................................................71

4.16 Calculated free energy \( (F) \) of gold nanoparticles described by the Sutton-Chen potential with Pawluk’s parameterization (SCP) as a function of temperature ..............................................72

4.17 Calculated internal energy \( (U) \) of gold nanoparticles described by the Sutton-Chen potential with Pawluk’s parameterization (SCP) as a function of temperature ..............................................72

4.18 Calculated entropy \( (S) \) of gold nanoparticles described by the Sutton-Chen potential with Pawluk’s parameterization (SCP) as a function of temperature ..............................................73

4.19 Calculated specific heat \( (C_v) \) of gold nanoparticles described by the Sutton-Chen potential with Pawluk’s parameterization (SCP) as a function of temperature ..............................................73

4.20 Calculated free energy \( (F) \) of gold nanoparticles described by the Quantum Sutton-Chen potential (Q-SC) as a function of temperature .................................................................74

4.21 Calculated internal energy \( (U) \) of gold nanoparticles described by the Quantum Sutton-Chen potential (Q-SC) as a function of temperature .................................................................74

4.22 Calculated entropy \( (S) \) of gold nanoparticles described by the Quantum Sutton-Chen potential (Q-SC) as a function of temperature .................................................................75
4.23 Calculated specific heat ($C_v$) of gold nanoparticles described by the Quantum Sutton-Chen potential (Q-SC) as a function of temperature .................................................................75

4.24 Calculated free energy ($F$) of gold nanoparticles described by the Embedded Atom Method (EAM) by Cai and Ye as a function of temperature ..........................................................76

4.25 Calculated internal energy ($U$) of gold nanoparticles described by the Embedded Atom Method (EAM) by Cai and Ye as a function of temperature ...............................................................76

4.26 Calculated entropy ($S$) of gold nanoparticles described by the Embedded Atom Method (EAM) by Cai and Ye as a function of temperature ...............................................................77

4.27 Calculated specific heat ($C_v$) of gold nanoparticles described by the Embedded Atom Method (EAM) by Cai and Ye as a function of temperature ...............................................................77

4.28 Calculated free energy ($F$) of gold nanoparticles described by the empirical potential for gold proposed by Olivier (POT) as a function of temperature ....................................................78

4.29 Calculated internal energy ($U$) of gold nanoparticles described by the empirical potential for gold proposed by Olivier (POT) as a function of temperature ....................................................78

4.30 Calculated entropy ($S$) of gold nanoparticles described by the empirical potential for gold proposed by Olivier (POT) as a function of temperature ....................................................79

4.31 Calculated specific heat ($C_v$) of gold nanoparticles described by the empirical potential for gold proposed by Olivier (POT) as a function of temperature ....................................................79

4.32 Melting point of gold nanoparticles. Solid line: theory (the modified Pawlow relation (equation (1.2))) [69], black squares: experimental [69, 75], dotted line: the least square fit of the experimental data, pink diamonds: MD simulation with MEAM [71], blue diamonds: MD simulation with EAM [72]. Circles: prediction of current study (empty: prediction with the LJ potential, sky-blue: prediction with the Gupta potential, navy: prediction with the SC potential, orange: prediction
4.33 Melting point of gold nanoparticles predicted by the Lennard-Jones potential (LJ) compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Lennard-Jones potential, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data .........................................................86

4.34 Melting point of gold nanoparticles predicted by the Gupta potential compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Gupta potential, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data .........................................................87

4.35 Melting point of gold nanoparticles predicted by the Sutton-Chen potential (SC) compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Sutton-Chen potential, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data.................................88

4.36 Melting point of gold nanoparticles predicted by the Sutton-Chen potential with Pawluk’s parameterization for small clusters (SCP) compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Sutton-Chen potential with Pawluk’s parameterization, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data ..........................................................89
4.37 Melting point of gold nanoparticles predicted by the Quantum Sutton-Chen potential (Q-SC) compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Quantum Sutton-Chen potential, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data .................................................................90

4.38 Melting point of gold nanoparticles predicted by the Embedded Atom Model (EAM) by Cai and Ye compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Embedded Atom Model (EAM) by Cai and Ye, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data .................................91
CHAPTER I

METALLIC NANOPARTICLES

Nanoscience and nanotechnology have become an intrinsic part of our daily lives. From medicines to electronic devices, they have enriched human life in various ways. Metal oxides, carbon nanotubes (CNT), gold and silver nanoparticles have been extensively used in the field of biomedicine. For example, ferric oxides ($\text{Fe}_2\text{O}_3$, $\text{Fe}_3\text{O}_4$) have been used as MRI contrast agents, carbon nanotubes (CNT) have been used in gene delivery and biosensors, and silver nanoparticles have been used in antibacterial creams and powders [1]. The wide applications of nanotechnology continue in electronics. Many metal, silica, and semiconductor nanoparticles are key components in electronics and computers. Copper and aluminum nanoparticles have improved the performance of electronic circuits, silica nanoparticles have been used as coatings and joining materials for optical fibers, and some semiconductor nanoparticles have been used in optoelectronic devices after doped with rare-earth materials [1]. Titanium oxides and zinc oxides in sunscreens and cosmetic products [1], nanoclays and carbon nanotubes (CNT) as electrodes for fuel cells [2, 3], and gold nanoparticles as a drug delivery agent [4] can also be an outstanding example of the wide application of nanoscience and nanotechnology.
Nanoparticles

Nanotechnology is the broad field of study of the behavior of matter on the nanometer length scale, and tries to understand, control, and manipulate it. Nanoparticles occupy a very important place in nanotechnology since they serve as the fundamental building blocks of the field. The term ‘nanoparticle’ refers to an object of size in the range of 1 to 100 nm at least in one of the three dimensions [1]. Nanoparticles can be made of metals, metal oxides, semiconductors, polymers, carbons, organics, or biomolecules. This is the basic criterion for the classification of nanoparticles, i.e., the chemical nature of the material that composes the nanoparticles. Nanoparticles can also be classified into nanospheres, nanorods, nanotubes, nanowires, nanofibers, etc. according to their shapes [1, 5, 6]. Nanoparticles with crystalline structure are called “nanocrystals” [6]. However, nanoparticles with multiply twinned structure (e.g. decahedron or icosahedron) shall not be called “nanocrystals” because they have no crystalline unit cells. A cluster of nanoparticles is referred to as “nanocluster” [6]. An array of carbon nanotubes (CNT) can also be called a “nanocluster”, while a single carbon nanotube (CNT) is just a nanoparticle. Nanoparticles smaller than 1 nanometer are often referred to as “clusters” since they show no crystalline structure and seem more like aggregates of atoms [6]. In this dissertation, the term “clusters” will often appear to replace the term “nanoparticle” since the size of the nanoparticles in this study is less than 2 nm.

Nanoparticles possess unique physical and chemical properties which are significantly different from those of bulk materials and those expected from the simple extrapolation of bulk properties [1, 5]. This is the reason why the topic of nanoparticles
has attracted so many scientists and researchers and opened up the possibility for numerous applications. The unique properties of nanoparticles can be explained by the high surface area per unit volume, the high proportion of surface atoms and near-surface atoms, and quantum effects that nanoparticles have [1]. In fact, these three characteristics of nanoparticles arise from one feature: the nanometer length scale typical of the object. It is quite clear that the surface area-to-volume ratio and the proportion of surface atoms increase with a decrease in size. The quantum effects in nanoparticles will be discussed in detail in the next section for the case of metal nanoparticles.

1.2 Metallic nanoparticles

Metallic nanoparticles are of great scientific interest due to their peculiar optical [7] and electrical [8] properties. These unique physical properties originate from their finite size. Since the size of metal nanoparticles is only a few or tens of nanometers by definition, the movement of their electrons is confined in three-dimensional space. Thus, they exhibit discrete energy levels. If a physical system exhibits discrete energy levels, it is best described by the laws of Quantum Mechanics [6]. Therefore, metal nanoparticles show properties and behaviors that are better described by Quantum Mechanics and cannot be explained by the laws of Classical Physics.
One example of the extraordinary properties of metal nanoparticles is that the color they display depends on their chemistry, size, and shape. The dependence can be seen in Figure 1.1, where gold and silver nanoparticles of different shapes and sizes display various colors. This is attributed to a phenomenon known as Surface Plasmon Resonance (SPR). Surface Plasmon Resonance (SPR) is a phenomenon that occurs in selected sizes of semiconductor nanoparticles and metal nanoparticles (e.g., Quantum Dots) [5, 6]. It refers to the oscillation of electrons on the surface of metal nanoparticles caused by the interaction between light and confined electrons of the nanoparticles. The process is schematically shown in Figure 1.2.
When Surface Plasmon Resonance (SPR) occurs, nanoparticles absorb a specific wavelength from the light that matches the frequency of the oscillation, and display a color. The frequency of the oscillation depends on the chemistry of the material, the size of the nanoparticle, the shape of the nanoparticle, the surrounding medium, and even on the temperature [10-12]. This is the reason why metal nanoparticles of different properties display different colors. For aqueous solutions of gold nanoparticles of diameter of 9, 15, 22, 48, and 99 nm, the maximum absorbance has been observed at the wavelengths 517, 520, 521, 533, and 575 nm, respectively. However, Surface Plasmon Resonance (SPR) was absent for gold nanoparticles with diameter less than 2 nm, as well as for bulk gold [13]. Surface Plasmon Resonance (SPR) has been exploited in biosensor applications [13, 14] and for resolution enhancement of spectroscopic measurements [13, 15]. For instance, gold nanoparticles have been used for DNA sensing. Gold
nanoparticles form conjugates with DNA strands and display a color that is different from that of the free gold nanoparticles. In the presence of the complementary DNA, the gold nanoparticle-DNA aggregates decompose and show a change in color [14].

Traditionally metal nanoparticles have been used as catalysts because of their high surface/volume ratio. For instance, gold clusters with diameters smaller than 6 nm have been used as a very effective catalyst for oxidation of carbon monoxide [16]. Recently, significant efforts have been made to use metal nanoparticles in biological applications. A typical example of this is gold nanoparticles used in drug delivery and biosensors. Gold nanoparticles are known to have excellent biocompatibility and low inherent toxicity. Thus they promise high expectation for possible applications in medical devices. A complex composed of gold and iron oxide nanoparticle can work as a platin delivery agent [17]. Platin is a therapeutic agent for breast cancer. The complex is able to detect a breast cancer cell and release the therapeutic agent in it. A case of gold nanoparticles in biosensor application was mentioned in the previous paragraph. Here is another example in which gold nanoparticles are used to detect a protein of interest [18]. In this example, magnetic microparticles with antibodies bind the target protein first and then gold nanoparticles encoded with DNA that is specific to the target protein and antibodies surround the target protein. The protein can be separated and identified after the magnetic microparticles separation and the DNA dehybridization process. The dominance of gold over other metals in medical and biological application is attributed to the low inherent toxicity, excellent biocompatibility, and non-immunogenicity of it.

The most valuable and fascinating properties of metal nanoparticles are their optical properties. Various shapes of gold and silver nanoparticles have been used in the
field of optics [7, 19]. Gold nanoparticles functionalized with chromophores have diverse applications along with the choice of the chromophore. For instance, gold nanoparticles linked with fluorophores can be used in photocurrent generation or fluorescent display devices [7]. Gold nanoparticles can even bind and release amino acids depending on the presence of the light when linked with appropriate chromophores [7].

1.3 Structural properties of metal nanoparticles

Among the various properties of metal nanoparticles, the most studied properties have been the structural properties [20]. It has been found, theoretically [21-23] and experimentally [24-26], that metal nanoparticles have crystallographic structures different from the ones of the bulk material. For example, gold nanoparticles have icosahedral or decahedral motifs depending on their size while bulk gold has face-centered cubic (fcc) structure [21, 22, 24]. These crystallographic structures are shown in Figure 1.3. They are energetically the most stable structures in their size ranges. In the case of gold, the icosahedral structure is considered to be the most stable structure for clusters smaller than 1 nm and the truncated-decahedral structure is considered to be the preferred one for clusters of size between 1 nm and 2 nm. For clusters larger than 2 nm, the truncated-octahedral structure (fcc) is the dominant shape [21, 22].
1.3.1 The Global Minimum Structure

The most stable structure or the global minimum structure of a metal nanoparticle is of great scientific importance. It refers to the structure that minimizes the total potential energy of a nanoparticle. In principle, the global minimum structure must be the structure at the temperature of absolute zero. It helps predict the stability of a nanoparticle of a certain structure under a given condition and understand the structural transitions that the nanoparticle would undergo. There have been extensive computational studies on finding the global minimum structure of a given metal nanoparticle [27]. In order to accurately predict the global minimum structure, an accurate force field for the system and a fast and efficient algorithm are required. In practice the force field is just a matter of choice although the results will strongly depend on it. The accuracy of a force field is determined by how well the resulting physical and chemical properties mimic the real system and constitutes a separate field of study. Since the accuracy of force fields shall comprise the main body of this dissertation, it will be discussed in Chapter IV. The global
optimization algorithm is of crucial importance in the study of the global minimum structure of metal nanoparticles. It is the engine that leads us to the structure of lowest energy. A brief summary of global optimization algorithms follows.

1.3.2 Global Optimization Algorithms

Conventional Monte Carlo simulation is the simplest, but still effective, method for global optimization. However, as the demand increased for the study of larger and more complicated systems that could not be studied by the conventional Monte Carlo simulation method, a number of novel global optimization algorithms were developed such as the simulated annealing method [28-31], genetic algorithms [32-34], the basin-hopping approach [35, 36], the dynamic lattice searching (DLS) method [37, 38], and others.

1.3.2.1 Simulated Annealing Method

The simulated annealing method attempts to find the global minimum structure by performing a series of conventional Monte Carlo simulations with the temperature decreasing slowly from a high temperature [28]. This method was successful in finding the global minimum structure of atomic clusters containing up to 155 atoms that are modeled with the Morse potential [31]. However, simulations using the simulated annealing methods could easily be trapped in a local minimum when the global minimum is located in a narrow valley of the potential energy surface (PES).
1.3.2.2 Genetic algorithms

Genetic algorithms maintain a number of clusters and create new clusters from the existing clusters by a selected rule, for instance, cutting the existing clusters with a random plane that passes through the center of mass and assembling the new clusters from two halves [32]. A new cluster may replace one of the existing clusters if its energy is lower than that of the existing cluster. The global minimum could hopefully be reached by repeating these steps. The global minima of Lennard-Jones clusters with atoms up to 75 were successfully located and new global minima of 88-atom and 98-atom Lennard-Jones clusters were found using the generic algorithm by Deaven [33]. The global minima of Lennard-Jones clusters with atoms up to 110 were successfully found by the genetic algorithm by Wolf [34].

1.3.2.3 Basin-hopping approach

The basin-hopping approach refers to a global optimization method that involves the transformation of the potential energy surface (PES) to a collection of discrete energy levels. Thus the energy can vary only in discrete steps. The “Monte Carlo plus energy minimization” procedure is a representative example of the basin-hopping approach. It successfully located all global minima of Lennard-Jones clusters with up to 110 atoms [35]. The merit of the basin-hopping approach is that it broadens the occupational probabilities of different structures. So, escaping from a large basin becomes easier and thus the sampling becomes more efficient. The genetic algorithms by Deaven [33] and Wolf [34], in fact, correspond to a basin-hopping approach since they use a minimization step on every move.
1.3.2.4 Dynamic lattice searching (DLS) method

The dynamic lattice searching (DLS) method starts from a randomly generated local minimum of a cluster and repetitively finds structures with lower energy by two kinds of move, i.e., lattice construction and lattice searching, followed by local minimization [37]. In lattice construction, possible location sites are searched for an added atom and in lattice searching, the atom located at the occupied lattice site with the highest energy is moved to the vacant lattice site with the lowest energy iteratively. This technique has a very high convergence speed since it utilizes monotonic evolution. However, it may converge at numerous local minima very easily depending on the choice of the initial local minimum structure. Multiple independent runs help resolve this problem. The global minima of Lennard-Jones clusters with atoms up to 309 were located by this method [37]. A modified version of this technique, called “dynamic lattice searching with constructed core (DLSc)”, shares the fundamentals of the dynamic lattice searching method, but starts with a predetermined core structure and optimizes only the atoms in the outer 2 shells of a cluster [38]. The DLSc is a biased method to facilitate the convergence. The convergence speed was improved compared to the DLS technique, but the probability of finding the global minimum became even lower. The DLSc successfully located the global minima of Lennard-Jones clusters with 100 – 200 atoms and 660 – 670 atoms [38] as well as silver clusters with atoms up to 160 described by the Gupta potential [39].
1.3.2.5 Global minima of metal nanoparticles

The global minima of metal nanoparticles described by Lennard-Jones potentials are known for clusters with up to 1000 atoms [29, 33, 35, 36, 40-54]. Most of the Lennard-Jones global minima are of icosahedral structure with several exceptions. The non-icosahedral global minima of Lennard-Jones clusters are shown in Table 1.1.

Table 1.1: Non-icosahedral global minima of Lennard-Jones clusters. N: number of atoms, TO: truncated-octahedral, mDh: Mark-decahedral, Td: tetrahedral [35-54].

<table>
<thead>
<tr>
<th>N</th>
<th>motif</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>TO</td>
</tr>
<tr>
<td>75-77</td>
<td>mDh</td>
</tr>
<tr>
<td>98</td>
<td>Td</td>
</tr>
<tr>
<td>102-104</td>
<td>mDh</td>
</tr>
<tr>
<td>185-192</td>
<td>mDh</td>
</tr>
<tr>
<td>236-238</td>
<td>mDh</td>
</tr>
<tr>
<td>650-664</td>
<td>mDh</td>
</tr>
<tr>
<td>682-689, 691</td>
<td>mDh</td>
</tr>
<tr>
<td>754-762</td>
<td>mDh</td>
</tr>
<tr>
<td>815-823</td>
<td>mDh</td>
</tr>
</tbody>
</table>

The global minima of metal nanoparticles described by more realistic force fields have also been reported for some face-centered cubic (fcc) metals and their alloys [39, 55-57]. Recent discoveries include the global minima of silver nanoparticles described by the Gupta potential [39], the global minima of gold nanoparticles described by the density functional theory (DFT) [56], and the global minima of gold-silver alloys described by the Gupta potential with the density functional theory (DFT) calculations [57]. The
structural motifs of the global minima found by the three studies are shown in Table 1.2. There is a website called the “Cambridge Cluster Database”. It contains a vast array of global optimization results [58].

Table 1.2: The global minima of gold (Au), silver (Ag), and gold-silver (Au-Ag) alloys. Ih: icosahedral, Oh: octahedral, TO: truncated-octahedral, mDh: Mark-decahedral, TC: truncated-cube, CO: cubeoctahedral. Oh, TO, cube, TC, and CO belong to the face-centered cubic (fcc) structure [39, 56, 57].

<table>
<thead>
<tr>
<th>N</th>
<th>Ag</th>
<th>Au</th>
<th>Au-Ag</th>
<th>N</th>
<th>Ag</th>
<th>Au</th>
<th>Au-Ag</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Ih</td>
<td>Ih</td>
<td></td>
<td>88</td>
<td>fcc</td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>amorphism</td>
<td></td>
<td>89</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>amorphism</td>
<td></td>
<td>90</td>
<td>fcc</td>
<td>Ih</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>amorphism</td>
<td></td>
<td>91</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>amorphism</td>
<td></td>
<td>92</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>amorphism</td>
<td>Oh</td>
<td>93</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>amorphism</td>
<td>Ih</td>
<td>94</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>amorphism</td>
<td>Ih</td>
<td>95</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>amorphism</td>
<td>Ih</td>
<td>96</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>amorphism</td>
<td>Ih</td>
<td>97</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>amorphism</td>
<td>Ih</td>
<td>98</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>amorphism</td>
<td>Ih</td>
<td>99</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>amorphism</td>
<td>Ih</td>
<td>100</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>amorphism</td>
<td>Ih</td>
<td>101</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>amorphism</td>
<td>Ih</td>
<td>102</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>amorphism</td>
<td>TC</td>
<td>103</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>amorphism</td>
<td>Ih</td>
<td>104</td>
<td>mDh</td>
<td>TC</td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>amorphism</td>
<td>Ih</td>
<td>105</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>amorphism</td>
<td>Ih</td>
<td>106</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>amorphism</td>
<td>cube</td>
<td>107</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>mDh</td>
<td></td>
<td>108</td>
<td>mDh</td>
<td>cube</td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>amorphism</td>
<td>Ih</td>
<td>109</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>amorphism</td>
<td>Ih</td>
<td>110</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>amorphism</td>
<td>Ih</td>
<td>111</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>amorphism</td>
<td>Ih</td>
<td>112</td>
<td>mDh</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>fcc</td>
<td>TO</td>
<td>TO</td>
<td>113</td>
<td>mDh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>fcc</td>
<td></td>
<td>114</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>fcc</td>
<td></td>
<td>115</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>mDh</td>
<td></td>
<td>116</td>
<td>mDh</td>
<td>CO</td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Ih</td>
<td></td>
<td>117</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>43</td>
<td>mDh</td>
<td></td>
<td>118</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>amorphism</td>
<td>Oh</td>
<td>lh</td>
<td>119</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
</tr>
<tr>
<td>45</td>
<td>amorphism</td>
<td></td>
<td>120</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>amorphism</td>
<td>lh</td>
<td>121</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>amorphism</td>
<td></td>
<td>122</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>amorphism</td>
<td>lh</td>
<td>123</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>Ih</td>
<td></td>
<td>124</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>fcc</td>
<td>mDh</td>
<td>125</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>Ih</td>
<td></td>
<td>126</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>Ih</td>
<td>lh</td>
<td>127</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>Ih</td>
<td></td>
<td>128</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>Ih</td>
<td>lh</td>
<td>129</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>Ih</td>
<td>Ih</td>
<td>130</td>
<td>mDh</td>
<td></td>
<td>lh</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>Ih</td>
<td>lh</td>
<td>131</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>mDh</td>
<td></td>
<td>132</td>
<td>mDh</td>
<td></td>
<td>lh</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>Ih</td>
<td>lh</td>
<td>133</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>mDh</td>
<td></td>
<td>134</td>
<td>mDh</td>
<td></td>
<td>lh</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Ih</td>
<td>mDh</td>
<td>135</td>
<td>mDh</td>
<td></td>
<td>mDh</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>Ih</td>
<td></td>
<td>136</td>
<td>mDh</td>
<td></td>
<td>lh</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>mDh</td>
<td>TC</td>
<td>lh</td>
<td>137</td>
<td>mDh</td>
<td></td>
<td>TO</td>
</tr>
<tr>
<td>63</td>
<td>mDh</td>
<td></td>
<td>138</td>
<td>mDh</td>
<td></td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>mDh</td>
<td>lh</td>
<td>139</td>
<td>mDh</td>
<td></td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>mDh</td>
<td></td>
<td>140</td>
<td>mDh</td>
<td></td>
<td>TO</td>
<td>lh</td>
</tr>
<tr>
<td>66</td>
<td>mDh</td>
<td>lh</td>
<td>141</td>
<td>mDh</td>
<td></td>
<td>TO</td>
<td>lh</td>
</tr>
<tr>
<td>67</td>
<td>mDh</td>
<td></td>
<td>142</td>
<td>mDh</td>
<td></td>
<td>Ih</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>mDh</td>
<td>mDh</td>
<td>143</td>
<td>mDh</td>
<td></td>
<td>Ih</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>mDh</td>
<td></td>
<td>144</td>
<td>mDh</td>
<td></td>
<td>Ih</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>mDh</td>
<td>mDh</td>
<td>145</td>
<td>mDh</td>
<td></td>
<td>Ih</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>mDh</td>
<td></td>
<td>146</td>
<td>mDh</td>
<td>Oh/mDh</td>
<td>Ih</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>mDh</td>
<td>mDh</td>
<td>147</td>
<td>lh</td>
<td>Ih</td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>mDh</td>
<td></td>
<td>148</td>
<td>mDh</td>
<td></td>
<td>Ih</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>mDh</td>
<td>mDh</td>
<td>149</td>
<td>fcc</td>
<td>Ih</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>mDh</td>
<td>mDh</td>
<td>150</td>
<td>mDh</td>
<td></td>
<td>Ih</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>mDh</td>
<td>mDh</td>
<td>151</td>
<td>fcc</td>
<td>Ih</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>mDh</td>
<td></td>
<td>152</td>
<td>fcc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>mDh</td>
<td>mDh</td>
<td>153</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>fcc</td>
<td>TO</td>
<td>154</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>fcc</td>
<td>mDh</td>
<td>155</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>mDh</td>
<td></td>
<td>156</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>mDh</td>
<td>mDh</td>
<td>157</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>fcc</td>
<td></td>
<td>158</td>
<td>fcc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>mDh</td>
<td>lh</td>
<td>159</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>fcc</td>
<td>Oh</td>
<td>160</td>
<td>mDh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>mDh</td>
<td>mDh</td>
<td>171</td>
<td>TC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>fcc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1.3.3 Structural transition of metal nanoparticles

The structural change of metal nanoparticles with increasing temperature has been investigated as extensively as has been the global minimum structure. Commonly known structural transitions of metal nanoparticles are melting (solid-to-liquid transition), solid-to-solid transition, and surface reconstruction. An illustrative example of the latter is the Mackay-to-anti-Mackay transition that takes place in the overlayer of a Lennard-Jones nanoparticle [59, 60]. The term ‘overlayer’ refers to the outermost shell of an atomic cluster. Mackay- and anti-Mackay- describe the way of atom packing in the overlayer of an icosahedral atomic cluster. The Mackay atom packing adds atoms in such a way that the added atoms will comprise the outer shell of the next complete icosahedron [61]. The anti-Mackay packing adds atoms tetrahedrally on the stacking fault sites and on the vertex sites. A schematic picture of Mackay- and anti-Mackay- atom packing is shown in Figure 1.4. The Mackay-to-anti-Mackay transition usually occurs at a temperature below the melting point and is attributed to the higher entropy of the anti-Mackay packing [59, 60].

Evidences of a solid-to-solid transition of metal nanoparticles have been found in the Lennard-Jones nanoparticles with a non-icosahedral global minimum structure [62-65]. For example, the Lennard-Jones nanoparticle with 38 atoms, known to have a truncated-octahedral global minimum structure, undergoes a structural transition to the Mackay icosahedral structure at a very low temperature [62, 63]. Subsequently, it experiences the Mackay-anti-Mackay transition at a little higher temperature and eventually melts to a liquid. The Lennard-Jones nanoparticle with 98 atoms shows a similar behavior. It undergoes a transition to the Mackay icosahedral structure from the
tetrahedral global minimum structure followed by the Mackay-anti-Mackay transition and the melting of the whole cluster. The Lennard-Jones nanoparticle with 75 atoms, too, exhibits a solid-to-solid transition, in which it changes its decahedral global minimum structure to the anti-Mackay icosahedral structure. The 75-atom Lennard-Jones nanoparticle does not undergo the Mackay-anti-Mackay surface reconstruction because after the solid-to-solid transition the structure is anti-Mackay icosahedral.

Figure 1.4: Illustration of Mackay- and anti-Mackay- atom packing

Another example of the sub-melting temperature structural transition has been observed in the process of melting of gold nanoparticles. It has been shown, through extensive molecular dynamics (MD) simulations, that gold nanoparticles would undergo a solid-to-solid transformation from the optimal structures, i.e., truncated decahedral or truncated octahedral, to the icosahedral structures at a temperature slightly below the melting temperature [66, 67]. The formed icosahedral structure is temporary and will
eventually melt to a liquid. This prediction of the icosahedral precursors to melting is interesting because it proposes a solid-to-solid transformation at such an elevated temperature.

1.4 Thermodynamic properties of metal nanoparticles

The thermodynamic aspects of metal nanoparticles have been studied as much as the structural properties. It has been found that the melting temperature of metal nanoparticles is size-dependent. For instance, the melting temperature of gold nanoparticles starts to decrease at a diameter of about 20 nm and drops more rapidly as the size decreases [68-73]. This is clearly shown in Figure 1.5. The melting temperature of a 1.6 nm gold nanoparticle was expected to be around 500 K while that of bulk gold is 1337 K [71].

Theoretical considerations about the depression of the melting point of metal nanoparticles have been developed along with experimental observations. Pawlow established a quantitative relationship between the particle size and the melting temperature based on the classical Kelvin equation [74]. For a nanoparticle of diameter $d$, the Pawlow equation is:

$$
\frac{T_m(d)}{T_m(\infty)} = 1 - \frac{4\nu_s^{2/3}}{L} \left( \gamma_s - \gamma_l \right) \frac{1}{d}
$$

(1.1)

where $L$ is the molar heat of fusion, $\nu_s$ and $\nu_l$ are the specific molar volume of the solid and liquid, $\gamma_s$ and $\gamma_l$ are the molar surface free energy of the solid and liquid, and $\infty$
signifies the bulk material. This equation is not quantitatively accurate but is still often used to estimate the size dependence of the melting temperature. Theoretical modifications have been applied to the original Pawlow equation and improved the agreement with experimental observations [75-78]. What is interesting is that they all predicted the inverse dependence of the melting temperature with particle size.

Figure 1.5: Melting point of gold nanoparticle as a function of its size. (○) experimental [69], (Δ) experimental [75], (■) MD simulation with MEAM [71], and (▼) MD simulation with EAM [72]. (Reprinted with permission from [71]. Copyright 2002 Elsevier)
Buffat and Borel carried out a thorough systematic study on the size dependence of the melting of gold nanoparticles and proposed their version of the modified Pawlow relation [69]:

$$\frac{T_m(d)}{T_m(\infty)} = 1 - \frac{4}{\rho_s L} \left( \gamma_s - \gamma_l \left( \frac{\rho_s}{\rho_l} \right)^{2/3} \right) \frac{1}{d}$$

(1.2)

where $\rho_s$ and $\rho_l$ are the density of the solid and liquid, and the rest of the symbols have the same meaning as those in equation (1.1). This equation has proved to give a very good description of the experimental observations, at least for gold nanoparticles [69, 76], and will be employed to compare the simulation results to experimental observations in Chapter IV.

The size dependence of the melting temperature of metal nanoparticles can also be explained with the fraction of the atoms on the surface. The properties of a surface atom are significantly different from those of an inner atom because their environments are different. In bulk gold, the physical properties are dominated by the inner atoms because the fraction of the surface atoms is negligible. Meanwhile, in a nanoparticle, the physical properties are severely affected by the surface atoms because they can no longer be neglected. For instance, 80% of the total atoms are surface atoms in a 1.5 nm gold nanoparticle. The atoms on the surface have lower bound energy compared to the inner atoms and thus melt at temperatures below the melting point of bulk gold.
1.5 Force Fields for metallic systems

The study of metal nanoparticles requires strong interplay of theory, experiments, and computer simulations. The relevance of the latter becomes more important in the study of small nanoparticles (<2~3nm) because experiments have limitations on the length scales that they can probe. The hub of any computational study is the potential energy function, also known as the force field, that describes the interactions between the components of the system because everything that comes out from the study depends on the potential energy function. Therefore, the choice of an accurate potential energy function is crucial in computational studies.

The force field is a collection of energy equations and their associated energy parameters used to describe the interactions among the constituents of the system. A wide variety of force fields has been developed to model different systems and is now available for use. The Consistence Valence Force Field (CVFF) [79] was developed to model small organic and bioorganic molecules and proved to describe such molecules very well. The Chemistry at HARvard Macromolecular Mechanics (CHARMM) [80] and the Assisted Model Building with Energy Refinement (AMBER) [81] are the most popularly used force fields to model the behavior of biomolecules including proteins, nucleic acids, lipids, and carbohydrates. These force fields are still being improved and updated. The most up-to-date version of CHARMM is CHARMM36 which came out on Aug 15, 2008 and that of AMBER is AMBER11 published on April 25, 2010.

There are many potential energy functions that can be applied to metallic systems. Density functional theory (DFT) [82] and ab initio calculations [83] are the most accurate
methods in describing metallic nanoparticles. They require calculations at the electronic level, i.e. quantum mechanical calculations, and provide the highest level of description of metallic systems. However, their use is limited to very small systems due to the demand for high computational power. Thus, many classical, atomistic potential energy functions have been commonly used in the study of metallic nanoparticles. Atomistic potential energy functions describe a system by the interactions among the constituent atoms and require calculations at the atomic level. Compared to the quantum mechanical calculation methods, atomistic potential energy functions require less computational power with less accuracy. But, they still provide good description of metallic nanoparticles.

Many atomistic potential energy functions have been popularly employed in the study of metallic nanoparticles. The most well-known potentials include the Lennard-Jones potential (LJ) [84], the Glue Model potential [85, 86], the Finnis-Sinclair potential (FS) [87], the Tight-Binding potential (TB) [88-91], the Gupta potential [90], the Sutton-Chen potential (SC) [92-94], the Quantum Sutton-Chen potential (Q-SC) [95-97], the Embedded Atom Method (EAM) [98-104], and the Modified Embedded Atom Method (MEAM) [105-107]. These potential energy functions are empirical or semi-empirical potentials developed to fit physical properties of the bulk such as lattice constants, elastic constants, cohesive energy, vacancy formation energy, sublimation energy, etc. Except for the Lennard-Jones potential, they are so-called many-body potentials, as they incorporate interactions that involve more than two atoms. It was shown that some difficulties associated with the use of pair potentials were satisfactorily overcome with the introduction of many-body interactions. For instance, the Cauchy discrepancy ($C_{12} \neq$...
C_{44}) and the vacancy formation energies were correctly reproduced with many-body potentials [87, 91]. The atomistic potential energy functions successfully describe many properties of bulk and large nanoparticles. However, deviations for small nanoparticles are expected since these potential energy functions were developed to fit bulk properties.

1.5.1 The Lennard-Jones potential (LJ) [84]

The Lennard-Jones potential (LJ) [84] is the simplest energy function that can be applied to metallic nanoparticles. It is a pair-wise potential that describes the interaction between two atoms based on their distance from one another. The mathematical form of the Lennard-Jones potential (LJ) is shown in equation (1.3):

\[
E = 4\varepsilon \left[ \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^{6} \right]
\]

where \(\varepsilon\) is the depth of the potential well, \(\sigma\) is the distance at which the potential becomes zero, and \(r\) is the distance between two atoms. The parameters \((\varepsilon\) and \(\sigma\)) are fitted to reproduce experimental data or quantum mechanical calculations. The Lennard-Jones potential consists of two terms, i.e., the repulsive term \((r^{-12} \text{ term})\) and the attractive term \((r^{-6} \text{ term})\). The form of the repulsive term was chosen exclusively for ease of computation and does not have theoretical justification since the repulsive force should depend on distance exponentially. The total energy of a system can be obtained by simple summation of the pair interactions over all possible pairs in the system, as expressed in equation (1.4).
The Lennard-Jones potential is particularly accurate for noble gases and long-range van der Waals systems, but not quite accurate for metallic systems. For instance, it gives a poor description of surface relaxations in metals [92]. The use of the Lennard-Jones potential in the study of metal nanoparticles is decreasing as more sophisticated potentials become available. It is, however, often employed in algorithm tests and works as a guideline for other potentials.

1.5.2 The Glue Model potential [85, 86]

The Glue Model potential [85, 86] was first proposed by Ercolessi in 1986. It introduced the concept of many-body interaction to better describe metallic cohesion. The general expression for the Glue Model potential for a system of N atoms is:

$$E = \sum_i \sum_{j<i} \Phi(r_{ij}) + \sum_i U(n_i)$$

(1.5)

where $\Phi(r)$ represents the pair interaction, $n_i$ is the coordination of atom $i$, and $U(n)$ associates an energy value to the coordination $n$. The latter term in equation (1.5) is the many-body term, or glue term, that brings the non-linear dependence of the energy of an atom upon its coordination. The coordination of atom $i$ is obtained by the superposition of contributions from the neighboring atoms as expressed in equation (1.6),

$$n_i = \sum_{j \neq i} \rho(r_{ij})$$

(1.6)
The function \( \rho(r) \) is an empirical function that decreases monotonically with the distance. The analytical forms for the three functions \( \Phi(r), U(n), \) and \( \rho(r) \) were chosen to be polynomials and the parameters were determined to fit the zero temperature lattice parameter, the cohesive energy, the bulk modulus, the surface energy, and the phonon dispersion of the bulk. The ranges of the functions \( \Phi(r) \) and \( \rho(r) \) were limited to the first neighbors in the function building process. The Glue Model potential provided a good description of the bulk, defect and surface properties of gold. In particular, the occurrence of reconstruction on the low-index surfaces of gold was satisfactorily predicted by the Glue Model potential. However, the potential was not successful in predicting the melting point of gold.

Many empirical or semi-empirical potentials including the Embedded Atom Method (EAM) and the Tight-Binding potential (TB) are particular cases of the general glue model expression in the sense that they share the same idea of having a pair interaction term and a many-body term. However, the theoretical backgrounds from which the potentials were derived are different and that is what distinguishes the potentials from each other along with the analytical forms of the functions.

1.5.3 The Finnis-Sinclair potential (FS) [87]

The Finnis-Sinclair potential (FS) [87] was proposed by Finnis and Sinclair in 1984 in an attempt to overcome the shortcomings of pair potentials in the description of metals. An example of the drawbacks of pair potentials is the prediction of the vacancy formation energy. The vacancy formation energy is always estimated to be about equal to the cohesive energy, whereas it is experimentally found to be about one-third of the cohesive energy.
energy. So, a simple model that incorporates the essential band character of metallic cohesion was proposed. Similarly to the Glue Model potential [85, 86], the Finnis-Sinclair potential includes a many-body term to describe the metallic cohesion. The functional form of the Finnis-Sinclair potential is shown in equation (1.7):

\[ E = \sum_i \sum_{j<i} V(r_{ij}) - A \sum_i f(\rho_i) \]  

(1.7)

where \( V(r) \) represents the repulsive pair interaction, \( A \) is a constant, \( \rho_i \) is the coordination of atom \( i \), and \( f(\rho) \) associates an energy value to \( \rho \). The \( \rho \) is a collective term and can be calculated as in equation (1.8)

\[ \rho_i = \sum_{j \neq i} \Phi(r_{ij}) \]  

(1.8)

where \( \Phi(r) \) is a function that describes the cohesive characteristic of metals.

The Finnis-Sinclair potential (FS) is based on the second moment approximation in tight-binding theory [88] and adopts the square root as the functional form for the many-body interaction \( f(\rho) \). The cohesive potential \( \Phi(r) \) was chosen to be in a parabolic form and the pair-potential \( V(r) \) took a form of quartic polynomial. Only the first neighbors and the second neighbors were considered in the development of these functions. The parameters of the Finnis-Sinclair potential were determined to correctly reproduce the cohesive energy, the equilibrium volume, and the three elastic constants of bulk. The Finnis-Sinclair potential correctly describes the relaxation at free surfaces of body-centered cubic (bcc) metals. It predicts inward relaxation of the surface layer which is in agreement with experimental observations. However, it is a short-ranged potential and thus fails to give an accurate description of long-range interactions. For instance, it
cannot distinguish the face-centered cubic (fcc) structure from the hexagonal close
packed (hcp) structure.

1.5.4 The Tight-Binding potential (TB) [88-91]

The Tight-Binding potential (TB) [88-91] is a semi-empirical potential for metals based
on the second-moment approximation of the tight-binding theory [88]. There are diverse
versions of Tight-Binding potential with different parameterizations and they are known
under different names to avoid confusion. The most popular Tight-Binding potentials are
the Gupta potential [90] and the RGL potential [91]. Since the Gupta potential will be
treated separately in the next section, the illustration of the Tight-Binding potential will
be based on the RGL potential.

The Tight-Binding potential (TB) consists of an attractive part and a repulsive
part as many other potentials do. The attractive term is derived from the second-moment
approximation of the tight-binding theory and the repulsive term is assumed to be a Born-
Mayer type. The mathematical form of the RGL potential is shown in equation (1.9)

\[
E = \sum_i \left( \sum_{j \neq i} A \exp[-p \left( \frac{r_{ij}}{r_0} - 1 \right)] - \xi \sqrt{\sum_{j \neq i} \exp[-q \left( \frac{r_{ij}}{r_0} - 1 \right)]} \right)
\]

(1.9)

where \(r_0\) is the first neighbor distance, \(r_{ij}\) is the distance between atoms \(i\) and \(j\), and \(A, p, \xi, q\)
and \(q\) are constants determined by fitting the experimental values of lattice parameter,
bulk modulus, elastic constant, and cohesive energy. The RGL potential provides a good
description of phonon dispersion relations, vacancy and interstitial formation energies,
and surfaces relaxations at zero temperature. Moreover, the root-mean-square
displacement and the thermal expansion show satisfactorily agreement with experimental values at low temperatures \( T < 0.65T_m \) for Cu and Ni. \( T_m \) defines the melting temperature. At high temperatures \( T > 0.65T_m \), however, the root-mean-square displacements were overestimated.

The main advantage of the Tight-Binding potential (TB) is the suitability for extension to higher order moment approximation which would lead to a better description of many physical properties. For instance, the stacking fault energy in face-centered cubic (fcc) or hexagonal close packed (hcp) transition metals was correctly reproduced with the sixth-moment approximation [106]. The stacking fault energy is the energy difference per unit area associated with a stacking fault, such as a small region of hexagonal close packed (hcp) crystal in a face-centered cubic (fcc) crystal [107]. The main shortcoming of the Tight-Binding potential also resides in its nature of using the second-moment approximation. The properties that depend on the moments of higher order, e.g. elastic constants [108], are very difficult to reproduce. This can be resolved, however, by introducing more fitting parameters as in the Finnis-Sinclair potential. The RGL potential has another drawback in that it does not provide a good description of long-range interactions. This is because the RGL potential takes into account only the first neighbors. A Tight-Binding potential that includes long-range interactions which is the Gupta potential [90] was developed later.

1.5.5 The Gupta potential [90]

The Gupta potential [90] is a kind of Tight-Binding potential developed by Cleri and Rosato. It is also known as the “long-range Tight-Binding potential” because it takes into
account long-range interactions by incorporating up to the fifth nearest neighbors. The Gupta potential shares the mathematical formula with the RGL potential (see equation (1.9)). However, the parameters in the formula are different from those of the RGL potential because the Gupta potential incorporates up to fifth nearest neighbors and the fitting is to the atomic volume, the elastic constants, and the cohesive energy at zero temperature.

Table 1.3: Parameters of the RGL potential [91] and of the Gupta potential [90] for face-centered cubic (fcc) metals.

<table>
<thead>
<tr>
<th></th>
<th>RGL</th>
<th>Gupta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A (eV)</td>
<td>ξ (eV)</td>
</tr>
<tr>
<td>Ni</td>
<td>0.137</td>
<td>1.76</td>
</tr>
<tr>
<td>Cu</td>
<td>0.099</td>
<td>1.35</td>
</tr>
<tr>
<td>Rh</td>
<td>0.097</td>
<td>2.00</td>
</tr>
<tr>
<td>Pd</td>
<td>0.168</td>
<td>1.72</td>
</tr>
<tr>
<td>Ag</td>
<td>0.123</td>
<td>1.28</td>
</tr>
<tr>
<td>Ir</td>
<td>0.144</td>
<td>2.50</td>
</tr>
<tr>
<td>Pt</td>
<td>0.234</td>
<td>2.50</td>
</tr>
<tr>
<td>Au</td>
<td>0.216</td>
<td>1.84</td>
</tr>
</tbody>
</table>

The Gupta potential successfully reproduces the phonon dispersion curves and the formation volumes and energies of simple point defects such as vacancy, self-interstitials, and dissociated Frenkel pair. A Frenkel pair is a type of point defect in a crystal in which an atom creates a vacancy in the crystal lattice and lodges in a nearby location that is usually not a lattice point. Furthermore, the extension of the interaction to a longer range provides good description of various thermodynamic properties up to the melting point,
including specific heat, thermal expansion coefficient, latent heat, enthalpy, and the
density of the liquid at the melting point.

1.5.6 The Sutton-Chen potential (SC) [92]

The Sutton-Chen potential (SC), proposed by Sutton and Chen in 1990 [92], is a Finnis-Sinclair potential (FS) with long-range interactions. It combines the superior description of short-range interactions in the Finnis-Sinclair potential with a van der Waals tail to provide more accurate description of the long-range interactions. The mathematical formula of the Sutton-Chen potential reads

\[
E = \varepsilon \left( \sum_i \sum_{j<i} V(r_{ij}) - c \sum_i \sqrt{\rho_i} \right)
\]

(1.10)

\[
V(r_{ij}) = \left( \frac{a}{r_{ij}} \right)^n
\]

(1.11)

\[
\rho_i = \sum_{j \neq i} \left( \frac{a}{r_{ij}} \right)^m
\]

(1.12)

where \( V(r) \) describes repulsive pair interactions, \( r_{ij} \) is the distance between atoms \( i \) and \( j \), \( \rho_i \) is the coordination of atom \( i \), and \( \varepsilon, c, a, m, \) and \( n \) are the fitting parameters. These parameters were determined so as to exactly fit the experimental values of cohesive energy and crystal lattice parameter. The Sutton-Chen potential successfully reproduces the bulk modulus and the elastic constants of various metals and their alloys. The parameters for metallic alloys are slightly different from those for pure metals [93]. The simplicity of the analytic formula of the Sutton-Chen potential makes the potential very versatile so that many variants with different sets of parameters have been proposed and
have proved to provide accurate description of many properties of metals and their alloys [94-97].

1.5.7 The Quantum Sutton-Chen potential (Q-SC) [95]

The Quantum Sutton-Chen potential (Q-SC), proposed by Kimura et al. in 1998 [95], is a modified Sutton-Chen potential that includes quantum corrections such as zero-point energy. It shares the analytic formula with the Sutton-Chen potential (SC) (see equation (1.10), (1.11), (1.12)). The parameters of the Quantum Sutton-Chen potential were determined to correctly reproduce the lattice parameter, cohesive energy, bulk modulus, elastic constants, vacancy formation energy, surface energy, and phonon dispersion behaviors. The parameters of the Sutton-Chen potential (SC) and the Quantum Sutton-Chen potential (Q-SC) for selected face-centered cubic (fcc) metals are shown in Table 1.4.

<table>
<thead>
<tr>
<th></th>
<th>Sutton-Chen potential (SC)</th>
<th>Quantum Sutton-Chen potential (Q-SC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ε</td>
<td>c</td>
</tr>
<tr>
<td>Ni</td>
<td>15.707</td>
<td>39.432</td>
</tr>
<tr>
<td>Cu</td>
<td>12.382</td>
<td>39.432</td>
</tr>
<tr>
<td>Ag</td>
<td>2.5415</td>
<td>144.41</td>
</tr>
<tr>
<td>Au</td>
<td>12.793</td>
<td>34.408</td>
</tr>
</tbody>
</table>

Table 1.4: Parameters of the Sutton-Chen potential (SC) and the Quantum Sutton-Chen potential (Q-SC) for several face-centered cubic (fcc) metals [92, 95]. [ε]= meV, [a]= Å, and c, m, and n are dimensionless.
The Quantum Sutton-Chen potential (Q-SC) provides an accurate description of various thermal and mechanical properties, e.g. the equation of states, thermal expansion, and specific heat, for face-centered cubic (fcc) metals (Ni, Cu, Ag, Ir, Rh, Pd, Au, Pt). Moreover, the agreement with experimental values is always equal or better for all properties except elastic constants, when compared to the Sutton-Chen potential (SC).

1.5.8 The Embedded Atom Method (EAM) [98, 99]

The Embedded Atom Method (EAM) [98, 99] is a semi-empirical many-body potential based on the local density approximation theory (LDA) [111, 112] for computing the total energy of a metallic system. The local density approximation (LDA) theory is a kind of density functional theory (DFT) that calculates the energy of a system from the exchange-correlation energy of the homogeneous electron gas [113]. In the scheme of the Embedded Atom Method (EAM), the total energy of a metallic system is described as a sum of pair repulsions and the energy required to embed the system into the electron gas caused by surrounding atoms. It can be expressed in a mathematical formula as in equation (1.13)

\[ E = \sum_i F_i(\rho_i) + \sum_{i<j} \Phi(r_{ij}) \]  

(1.13)

where \( F_i(\rho_i) \) is the embedding energy required to place atom \( i \) in an electron density of \( \rho_i \), \( \rho_i \) is the host electron density of atom \( i \), \( \Phi(r_{ij}) \) is the pair repulsion between atom \( i \) and \( j \), and \( r_{ij} \) is the distance between atom \( i \) and \( j \). The host electron density of atom \( i \), \( \rho_i \), can be
written as the sum of the electron density contributions from all other atoms as expressed in equation (1.14).

$$\rho_i = \sum_{j \neq i} f(r_{ij})$$

(1.14)

where \(f(r_{ij})\) describes the atomic electron density of atom \(j\) of separation \(r_{ij}\).

A number of functional forms for \(F(\rho)\), \(\Phi(r)\), and \(f(r)\) have been proposed in an attempt to provide a better description of metallic systems. In the original version of EAM (EAM-DB), by Daw and Baskes [98, 99], the atomic density \(f(r)\) was evaluated from Hartree-Fock calculations [114, 115], and the embedding function \(F(\rho)\) and pair repulsion function \(\Phi(r)\) were given in the form of natural splines that were obtained empirically. The non-analytical form of this EAM-DB model generates some inconvenience for calculations and limits the model from being extended to include an angular force. Another version of EAM (EAM-FBD), proposed by Foiles, Baskes, and Daw [100], takes the total energy to follow the equation of state of Rose et al. [116], and proposes an analytical form for the pair repulsion function. The EAM potential proposed by Adams, Foiles, and Wolfer (EAM-AFW) [101] uses the same functional forms of EAM-FBD with slightly different parameterization. Cai and Ye proposed an analytical EAM potential that includes long-range interactions [104]. In this model, the embedding function is of the universal form suggested by Banerjea and Smith [117], the electron density function is to be an exponentially decreasing function, and the pair repulsion function takes a form given by Rose et al. [116].

Although the details of the various versions of EAM models are different, they accurately describe many properties of metals including defects, liquids, surfaces, alloys,
mechanical properties, and hydrogen on surfaces. The advantage of the semi-empirical EAM models over purely empirical potentials is that they are easily extendable to more complicated systems, e.g. alloys, because they have a strong theoretical foundation. Among the various versions of EAM models, the one proposed by Cai and Ye [104] is the most attractive model because of its analytical functional form and the inclusion of long-range forces.

1.5.9 The modified Embedded Atom Method (MEAM) [105]

The modified Embedded Atom Method (MEAM), proposed by Baskes [105], is a specific version of Embedded Atom Method (EAM) with angular corrections. In the scheme of EAM, a linear superposition of spherically averaged atomic electron densities produces the total electron density as expressed in equation (1.14). This can be improved by adding gradient and higher-order corrections to the simple background density. The MEAM model basically shares the basic formula with the EAM model except the evaluation of the atomic electron density. The mathematical form for the calculation for the atomic electron densities in the MEAM model is very complicated and is not shown here.

In the development of the original MEAM model, only the first nearest neighbors were considered [105]. The original MEAM model successfully described the elastic behavior and simple defect properties of various metals. However, it could not address a number of problems like the structural instability, phase transitions, and incorrect surface reconstruction. An improved version of the MEAM model that takes into account the first- and second-nearest neighbors was developed by Lee, Shim, and Baskes in 2003 [108]. This MEAM model not only solved most of the problems that were associated
with the original MEAM model, but also provided a good description of the bulk properties, point defect properties, and thermal properties for face-centered cubic (fcc) metals.

There are more force fields for metallic systems than the ones introduced here. These include the EAM-LJ model [118] and the power-function series by Li [119, 120]. In addition, a new parameterization for the Lennard-Jones potential was proposed recently for selected metallic systems [121]. Meanwhile, quantum mechanical or first principle calculations are evolving quickly thanks to the rapidly growing computational power to reach for the ultimate goal of describing the real system more accurately.

1.6 Motivation of the study

As nanotechnology advances further, the sizes of the systems we have to control and manipulate will become smaller. The need for understanding the nature and properties of these small systems will be unavoidable for the next generation of nanotechnologists. Density functional theory or \textit{ab initio} calculations are the best choice for the computational study of metal nanoparticles, but they require huge computational power. Thus, its current use is limited to very small systems. For the Monte Carlo study of thermodynamic properties of metal nanoparticles of size of 1.3 nm, for instance, first principle calculations require too much time and they cannot be completed for several
years. The use of atomistic potential energy functions is a solution to this problem if the accuracy of the potential energy function is known to be good in the size range of interest.

This motivates the current study. The purpose of this dissertation is to explore if the currently available energy functions are accurate enough to describe the physical properties of metal nanoparticles. Since the accuracy of the atomistic potential energy functions is unknown for metal nanoparticles, the main goal of this dissertation will be to evaluate various atomistic potential energy functions to see if any of them provides an accurate description of the physical properties of metal nanoparticles of size smaller than 1.3 nm. We have selected gold among various types of known metal nanoparticles as the test case for this work because gold nanoparticles have been extensively investigated in the literature.

The basic rule for the selection of the force field to be employed in this work is to pick the ones that include long-range forces. Let us recall the review of the force fields. The various force fields can be grouped by their theoretical foundation and mathematical forms. The Finnis-Sinclair potential (FS) [87], the Sutton-Chen potential (SC) [92, 93], and the Quantum Sutton-Chen potential (Q-SC) [95] consist the first group. The Gupta potential [90] and the RGL potential [91] are a kind of the comprehensive Tight-Binding model (TB). The Embedded Atom Method (EAM) [98], the modified Embedded Atom Method (MEAM) [105], and the Glue Model potential [85] can be grouped together. The Sutton-Chen potential (SC) is a modified version of the Finnis-Sinclair potential (FS) that includes long-range interactions. Thus, the Finnis-Sinclair potential (FS) was excluded from our study. The RGL potential was excluded due to the inclusion of the Gupta potential in this research. The Embedded Atom Method (EAM) by Cai and Ye [104] was
selected as a representative of the glue model because the EAM is the most well-known and popular model among the various kinds of the glue model, and the model by Cai and Ye includes long-range forces. Unfortunately, the modified Embedded Atom Method (MEAM) [105] could not be studied in this work due to the difficulties in its implementation.

The potential energy functions selected for this work are the following ones:

- The Lennard-Jones potential (LJ) [84]
- The Lennard-Jones potential with Heinz’s parameterization (LJH) [121]
- The Gupta potential [90]
- The Sutton-Chen potential (SC) [92]
- The Sutton-Chen potential with Pawluk’s parameterization for small clusters (SCP) [94]
- The Quantum Sutton-Chen potential (Q-SC) [95]
- The Embedded Atom Method (EAM) by Cai and Ye [104]
- The empirical potential for gold proposed by Olivier and coworkers (POT) [92]
2.1 Fundamentals of Monte Carlo simulation [123]

Monte Carlo simulation is a computer simulation methodology that explores the phase space of a system in a time-independent and stochastic manner. It uses a sequence of random numbers to render and sample the phase space, from which diverse physical properties of the system can be extracted. Monte Carlo simulations have been applied to various problems in mathematics, statistical physics, or even in finance and business. For example, it was used to estimate the value of $\pi$ numerically. The main advantage of Monte Carlo simulation over many analytical techniques is that it is a simple and straightforward method that can provide an approximate solution to the problems encountered in complex systems.

In fact, Monte Carlo simulation refers to a group of algorithms that conduct a random sampling of a system. Regarding the Monte Carlo simulation in statistical mechanics, a region of phase space of a system is sampled from which certain properties of the system are estimated. For example, the equilibrium thermodynamic properties of a
gas in a box can be estimated by studying the diffusion of the gas molecules with Monte Carlo simulation. The accuracy of the estimation can be improved by running the simulation for longer times, i.e. having more samples. In the next sections a number of algorithms employed in Monte Carlo simulations will be introduced along with the discussion of their advantages and disadvantages.

2.2 Simple Sampling

The simplest method for performing a Monte Carlo simulation is known as “Simple Sampling”. Under the scheme of this technique, a series of random numbers is generated for even sampling of the phase space. Each individual number is selected entirely by chance and has the same probability of being chosen at any stage of the sampling process. This algorithm has been applied to many mathematical problems such as the numerical integration of complex multi-dimensional functions. The main disadvantage of this scheme is that it is inefficient for the study of physical systems.

2.3 Importance Sampling (Metropolis algorithm)

In order to overcome the inefficiency of the “Simple Sampling” in the study of physical systems, a smart sampling scheme was proposed. In this scheme, the sampling of the phase space of a system depends on a weight factor that enables more frequent sampling of important phases. The algorithm that performs this sampling scheme is known as the
“Metropolis algorithm” which is the most fundamental algorithm for the study of physical systems.

The weighted sampling of a system is done by rejecting the sampling of the unimportant phases. In the process of sampling, the transition from one phase to another one is accepted or rejected with an acceptance probability given in equation (2.1)

\[
p(A \rightarrow B) = \min\{1, \exp\left[\frac{(E_A - E_B)}{k_B T}\right]\}
\]

(2.1)

For instance, a transition from a higher energy state to a lower energy state, i.e., \( E_A > E_B \), is always accepted whereas a move to the opposite direction is accepted with a probability of a number between 0 and 1. This is schematically shown in Figure 2.1.

![Figure 2.1: Schematic picture of the energy landscape illustrating the concept of “Importance Sampling”.

"Importance Sampling".](image)
The mechanism of the Metropolis algorithm can be summarized as follows.

1. Select an initial state.

2. Find a new state.

3. Calculate the energy difference ($\Delta E$) between the new state and the initial state.

4. Generate a random number $r$ in the interval of $(0, 1)$.

5. Accept the move if $r > \exp(-\Delta E / k_B T)$. Otherwise reject the move.

6. Repeat 2 ~ 5.

Generally, the sampling of low energy states is difficult because of their low probability. The Metropolis algorithm imposes the acceptance rule in order to sample the low energy states more often. As a result, the sampling process becomes more efficient and the running time of the simulation is decreased.

However, the Metropolis algorithm has two significant drawbacks. The first one is that the system can be easily trapped in a local minimum at low temperatures. This is attributed to the nature of the algorithm that forces the system to low energy states and allows little chance of reverting to high energy states. There have been efforts to overcome this problem such as the Parallel Tempering method. In the Parallel Tempering method, many replicas of a system are run independently in multiple windows with different conditions, for instance, different temperatures. Two states with same energy in different windows are allowed to switch after a certain number of Monte Carlo steps in
order to sample the phase space more effectively and, thus help prevent the trapping in local minima.

The second drawback of the Metropolis algorithm is that it provides information of the system only for the condition at which the simulation was run. For instance, the information obtained at a certain temperature cannot be used for the study of the same system at different temperatures. Therefore, the Metropolis algorithm is not the optimal choice for the study of the temperature dependence of physical properties because multiple independent simulations at various temperatures have to be done.

A number of Monte Carlo algorithms can provide the information of a system under various conditions from a single simulation. These methods require a reweighting procedure to get the probability distribution at the desired condition from the probability distribution obtained from the simulation. A typical example is the multicanonical ensemble method [124] which estimates the Density of States and performs a random walk with a flat histogram. The multicanonical algorithm has served as an effective tool for the study of systems with a rough energy surface as has been proved to be very efficient in studying first-order phase transitions [125].

2.4 The Wang-Landau algorithm and the Density of States [126, 127]

A novel algorithm based on the multicanonical algorithm was proposed by Wang and Landau in 2001. This algorithm accurately estimates the Density of States $g(E)$ of a system by a random walk in energy space. The Density of States is the number of states at each energy level of a system. It is a fundamental characteristic from which many
physical properties may be derived including the thermodynamic properties. The Density of States obtained from the Wang-Landau algorithm is usually rescaled so that there is only one global energy minimum for every system. This can be expressed as $g(E_{\min}) = 1$ or $\ln g(E_{\min}) = 0$.

The basic idea behind the Wang-Landau algorithm is that a random walk in energy space with a probability proportional to the inverse of the Density of States should yield a flat histogram for the energy distribution. The transition probability from a state with energy $E_1$ to another state with energy $E_2$ is given by

$$p(E_1 \to E_2) = \min \left[ \frac{g(E_1)}{g(E_2)}, 1 \right] \quad (2.2)$$

The protocol for the Wang-Landau algorithm can be summarized as follows.

1. Set all Density of States $g(E)$ for all energy levels $E$ to $g(E) = 1$ and the histogram $H(E)$ to $H(E) = 0$.

2. Perform a random walk with the transition probability given in equation (2.2).

3. Each time an energy level $E$ is visited, update the corresponding Density of States with a modification factor $f > 1$ as $g(E) \to f \cdot g(E)$ and the histogram as $H(E) = H(E) + 1$. The initial value of the modification factor is commonly set to $f = f_0 = e \approx 2.71828...$.
4. Keep running the simulation and modifying the Density of States until the histogram becomes “flat”. The criterion for the flatness of the histogram may be set as all the entries in the histogram should be within 10% or 20% of the average of them.

5. If the histogram is flat, the Density of States converges to the true value with an error proportional to \( \ln(f) \). Then, reset the histogram for all energy levels \( H(E) = 0 \) and start the next cycle of random walks with a reduced modification factor \( f_i \) which is set by a monotonically decreasing function such as \( f_i = \sqrt{f_o} \) or \( f_i = \frac{1}{3} f_o \).

6. Continue the random walk until the histogram becomes flat again at which point the modification factor is reduced as before \( f_{i+1} = \sqrt{f_i} \) and the histogram is reset to zero again.

7. Stop the simulation when the modification factor becomes smaller than a predefined value such as \( f_{final} = \exp(10^{-8}) \approx 1.00000001 \).

The Wang-Landau algorithm overcomes the two shortcomings of the Metropolis algorithm, i.e., it is not trapped in local energy minima and it can provide the thermodynamic information of a system over a wide range of temperatures from a single run of simulation. With these advantages, the Wang-Landau algorithm has been applied to a broad range of problems in biophysics [128-130].
2.5 Wang-Landau algorithm with global updates [131]

A modified version of the Wang-Landau algorithm was proposed by Zhou. This algorithm saves a large number of Monte Carlo steps by a manipulation of the Density of States through global updates. The Monte Carlo steps needed to reach a flat histogram in the Wang-Landau algorithm generally increase as the modification factor becomes smaller and sometimes it results in a very long running time. This is simply because finding a low energy state becomes more difficult as the Density of States converges to the true Density of States. In this modified version of Wang-Landau algorithm, the Density of States is modified in order to force the random walk to explore low energy states. In addition, the modification factor is not reduced to an extremely small number. The mechanism of this algorithm can be summarized as follows.

1. The whole process consists of two stages, i.e., the initial accumulation stage and the refining stage.

2. In the initial accumulation stage, a random walk is performed as in the original Wang-Landau method with the acceptance probability

\[ p(i \rightarrow f) = \min \{1, \exp \left[ \ln \alpha [w(x_i) - w(x_f)] \right] \}. \]

\( w(x) = \ln g(x) \) is the Density of States and \( \alpha \) is a constant.

3. The Density of States is updated by

\[ w(x) \rightarrow w(x) + \gamma k \left( \frac{x - x_0}{\delta} \right) \] for an accepted move when the random walker arrives at state \( x \). The \( k(x) \) is a positive continuous kernel function such as a Gaussian kernel

\[ k(x) = \exp(-x^2) \]. \( \gamma \) and \( \delta \) are constants to scale the kernel function.
3. When \( w(x) \) is larger than a threshold that we call \( \omega \) for some \( x \), the global update is applied by
\[
w(x) \rightarrow w(x) + \kappa \exp\left[\frac{-\lambda}{w(x) - \omega}\right] \Theta(w(x) - \omega)
\]
and the accumulation continues with local updates. \( \kappa \) and \( \lambda \) are constants and \( \Theta \) is the Heaviside step function.

4. Once the Density of States resumes a uniform growth over the explored region, another global update is applied and the accumulation continues with local updates.

5. When the entire area of interest is explored, the accumulation continues with local updates only until a uniform growth over the entire area of interest is observed. This process is called the refining stage.

This algorithm was reported to save 90\% of the simulation time compared to the original Wang-Landau method while maintaining the advantageous feature of it. Due to such advantages, this algorithm was chosen to be employed in this study. However, a minor modification had to be applied to the algorithm in order to improve the accuracy. This will be discussed in section 3.1.
CHAPTER III

SIMULATION PROTOCOL

In this Chapter, I discuss the algorithm employed followed by the mathematical forms of the potential energy functions and a detailed description of how gold nanoparticles are modeled in the Monte Carlo simulation.

3.1 The modified Wang-Landau algorithm

Monte Carlo simulations were carried out on gold nanoparticles of various sizes (or numbers of atoms) using a modified version of the Wang-Landau (WL) algorithm. As discussed in section 2.5, the modified Wang-Landau algorithm of Zhou [131] proved to be much faster than the original one. Hence, it was employed in this study. The test of the algorithm, however, showed that the modified Wang-Landau algorithm by Zhou did not yield the converged Density of States (DOS) for gold nanoparticles. Knowing that the convergence of the Density of States is proportional to the modification factor $f$, the most suspicious part of the failed algorithm was that the simulation finished with one refining stage with the $\gamma$ of $1+10^{-4}$ as taken from the reference. The $\gamma$ in the modified algorithm plays a similar role as the modification factor $f$ in the original algorithm. So,
additional refining stages were added with the $\gamma$ that was gradually reduced from $1 \times 10^{-4}$ to $1 \times 10^{-8}$ which is the value adopted in the original Wang-Landau algorithm. The $\gamma$ was reduced using the cubic-root. We were able to obtain the converged Density of States for gold nanoparticles with the final $\gamma$ of $1 \times 10^{-8}$ as shown in Figure 3.1. The specific heat ($C_v$) curves in Figure 3.2 clearly show that the $\gamma$ must be reduced to $1 \times 10^{-8}$ in order to get correct results. Thus, in this study we employed the modified version of the Wang-Landau algorithm by Zhou that includes the additional refining stages with final $\gamma$ of $1 \times 10^{-8}$.

![Figure 3.1: Density of States (DOS) of 55-atom Lennard-Jones cluster estimated with different values of final modification factor (f).](image-url)
Figure 3.2: Calculated heat capacity (Cv) for the 55-atom Lennard-Jones cluster estimated with different values of the final modification factor (f).

3.2 Force Fields for gold nanoparticles

The various force fields employed in this study were presented at the end of Chapter I. A brief review of them will be provided, if not stated in Chapter I, and their mathematical forms and parameterizations will be discussed in this section.
3.2.1 The Lennard-Jones potential (LJ) [84]

\[
E_{\text{total}} = 4\varepsilon \sum_i \sum_{j<i} \left[ \left( \frac{\sigma}{r_{ij}} \right)^{12} - \left( \frac{\sigma}{r_{ij}} \right)^6 \right]
\] (1.4)

\(\varepsilon = 5.29\) kcal/mol and \(\sigma = 2.644\) Å.

3.2.2 The Lennard-Jones potential with Heinz’s parameterization (LJH) [121]

A new parameterization for the Lennard-Jones potential was proposed by Professor Heinz. With the new parameters, the Lennard-Jones potential was reported to reproduce the densities, surface tensions, interface tensions, and mechanical properties of several face-centered cubic (fcc) metals including gold. \(\varepsilon = 5.29\) kcal/mol and \(\sigma = 2.629\) Å.

3.2.3 The Gupta potential [90]

\[
E = \sum_i \left\{ \sum_{j\neq i} A \exp[-p \left( \frac{r_{ij}}{r_0} - 1 \right)] - \xi \sqrt{\sum_{j\neq i} \exp[-2q \left( \frac{r_{ij}}{r_0} - 1 \right)]} \right\}
\] (1.9)

\(A = 0.2061\) eV, \(\xi = 1.790\) eV, \(p = 10.229\), \(q = 4.036\), \(r_0 = 2.885\) Å.
3.2.4 The Sutton-Chen potential (SC) [92]

\[
E = \varepsilon \left[ \sum_i \sum_{j<i} V(r_{ij}) - c \sum_i \sqrt[3]{\rho_i} \right]
\]  \hspace{1cm} (1.10)

\[
V(r_{ij}) = \left( \frac{a}{r_{ij}} \right)^n
\]  \hspace{1cm} (1.11)

\[
\rho_i = \sum_{j \neq i} \left( \frac{a}{r_{ij}} \right)^m
\]  \hspace{1cm} (1.12)

\[\varepsilon = 0.012793 \text{ eV}, \ c = 34.408, \ a = 4.08 \ \text{Å}, \ m = 8, \ n = 10.\]

3.2.5 The Sutton-Chen potential with Pawluk’s parameterization for small clusters (SCP) [94]

Pawluk et al. proposed a new parameterization of Sutton-Chen potential (SC) for small metal clusters. The new parameters were obtained by fitting the binding energy of dimers, trimers, 8-atom, and 13-atom cluster to the Density Functional Theory (DFT) calculations. The size of gold nanoparticles in this study ranges from 10 atoms to 150 atoms. This model was employed to see if it could be extended to larger systems. \[\varepsilon = 0.0064582 \text{ eV}, \ c = 33.857, \ a = 4.08 \ \text{Å}, \ m = 8, \ n = 10.\]
3.2.6 The Quantum Sutton-Chen potential (Q-SC) [95]

Although this model includes the quantum corrections, it shares the functional forms with the classical Sutton-Chen potential (SC). $\varepsilon = 0.0078052$ eV, $c = 53.581$, $a = 4.0651$ Å, $m = 8$, $n = 11$.

3.2.7 The Embedded Atom Method (EAM) by Cai and Ye [104]

The general equations for the EAM potential were given in Chapter I. The specific functional forms for the atomic electron density, the embedding energy, and the pair repulsion are given here along with the parameterizations.

\[
E = \sum_i F_i(\rho_i) + \sum_{ij} \Phi(r_{ij}) \tag{1.13}
\]

\[
\rho_i = \sum_{j \neq i} f(r_{ij}) \tag{1.14}
\]

\[
f(r) = f_e \exp[-\chi(r - r_e)] \tag{3.1}
\]

\[
F(\rho) = -F_0 \left[ 1 - \ln \left( \frac{\rho}{\rho_e} \right)^n \right] \left( \frac{\rho}{\rho_e} \right)^n + F_i \left( \frac{\rho}{\rho_e} \right) \tag{3.2}
\]

\[
\Phi(r) = -\alpha \left[ 1 + \beta \left( \frac{r}{r_a} - 1 \right) \right] \exp \left[ -\beta \left( \frac{r}{r_a} - 1 \right) \right] \tag{3.3}
\]

$f_e = 1$, $r_e = 2.885$ Å, $\chi = 4.6$ Å$^{-1}$, $F_0 = 3.03$ eV, $F_1 = 0.3681$ eV, $n = 0.5$, $\rho_e = 12$, $\alpha = 0.5317$ eV, $\beta = 1.9324$, $r_a = 1.2714$ Å.
3.2.8  The empirical potential for gold proposed by Olivier and coworkers (POT) [92]

This empirical potential was developed exclusively for gold based on the Tight-Binding model. It includes angular corrections that are necessary to describe the directionality effects in the Au-Au interactions. The validity of the model was established by fitting to the Perdew-Wang exchange-correlation functional (DF/PW91) [132].

In this model the total energy is expressed as a sum of spherical, dipolar, and dipole-quadrupole contributions as seen in (3.4). Each contribution can be independently calculated using equation (3.5), (3.6), and (3.7), respectively. The parameters and the coefficients for the model are given in Table 3.1.

\[
E = \sum_i \left( E_i^{sph} + E_i^{dip} + E_i^{dip-quad} \right) 
\]  

(3.4)

\[
E_i^{sph} = A_i \sum_{j \neq i} \exp\left[ -p \left( \frac{r_{ij}}{r_0} - 1 \right) \right] - \xi_i \sum_{j \neq i} \exp\left[ -2q \left( \frac{r_{ij}}{r_0} - 1 \right) \right] 
\]  

(3.5)

\[
E_i^{dip} = D_i \cdot \sum_{\sigma=x,y,z} \left( \sum_{j \neq i} P(r_{ij}; R_M) \frac{r_{ij}^\sigma}{r_{ij}} \right)^2 
\]  

(3.6)

\[
E_i^{dip-quad} = DQ_i \cdot dipole \cdot \left\{ \sum_{\sigma=x,y,z} \sum_{\sigma=x,y,z} \left[ \sum_{j \neq i} \exp\left[ -\alpha_q \left( \frac{r_{ij}}{r_0} - 1 \right) \right] \cdot \frac{r_{ij}^\sigma r_{ij}^{\sigma'}}{r_{ij}^2} \right] \right\}^2 
\]

\[- \frac{1}{3} \left[ \sum_{j \neq i} \exp\left[ -\alpha_q \left( \frac{r_{ij}}{r_0} - 1 \right) \right] \right]^2 \} \]  

(3.7)

\[
A_i = A_i^{(0)} + A_i^{(1)} c_i + A_i^{(2)} c_i^2 + A_i^{(3)} c_i^3 
\]  

(3.8)

\[
\xi_i = \xi_i^{(0)} + \xi_i^{(1)} d_i + \xi_i^{(2)} d_i^2 + \xi_i^{(3)} d_i^3 
\]  

(3.9)
\[ c_i = \sum_{j \neq i} f(r_{ij}; \alpha_c, R_i) \]  \hspace{1cm} (3.10)\\
\[ d_i = \sum_{j \neq i} f(r_{ij}; \alpha_d, R_d) \]  \hspace{1cm} (3.11)\\
\[ f(r; \alpha, R) = \begin{cases} 
[1 + \exp(\alpha(r - R))]^{-1}, & r \leq R \\
0.5 \cdot \exp[-\alpha(r - R)/2], & r \geq R 
\end{cases} \] \hspace{1cm} (3.12)\\
\[ D_i = D_i^{(0)} + D_i^{(1)} e_i + D_i^{(2)} e_i^2 + D_i^{(3)} e_i^3 \] \hspace{1cm} (3.13)\\
\[ DQ_i = DQ_i^{(0)} + DQ_i^{(1)} e_i + DQ_i^{(2)} e_i^2 + DQ_i^{(3)} e_i^3 \] \hspace{1cm} (3.14)\\
\[ e_i = \sum_{j \neq i} f(r_{ij}; \alpha_e, R_i) \] \hspace{1cm} (3.15)\\
\[ P(r; R_M) = \begin{cases} 
 p_i^{(2)} (r - R_M)^2 + p_i^{(3)} (r - R_M)^3 + p_i^{(4)} (r - R_M)^4 & \text{if } r \leq R_M \\
 p_i^{(5)} (r - R_M)^5 + p_i^{(6)} (r - R_M)^6 & \text{if } r \geq R_M 
\end{cases} \] \hspace{1cm} (3.16)
Table 3.1: Parameters and coefficients for the POT potential. $R_c$, $R_d$, $R_e$, and $R_M$ are in Å and the others are dimensionless [92].

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
<th>parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>12.40566</td>
<td>$q$</td>
<td>3.75564</td>
</tr>
<tr>
<td>$\alpha_c$</td>
<td>8.2366</td>
<td>$R_c$</td>
<td>3.315329</td>
</tr>
<tr>
<td>$\alpha_d$</td>
<td>4.436801</td>
<td>$R_d$</td>
<td>3.864539</td>
</tr>
<tr>
<td>$A^{(0)}$</td>
<td>0.197011</td>
<td>$\xi^{(0)}$</td>
<td>1.496829</td>
</tr>
<tr>
<td>$A^{(1)}$</td>
<td>-0.033297</td>
<td>$\xi^{(1)}$</td>
<td>0.026087</td>
</tr>
<tr>
<td>$A^{(2)}$</td>
<td>0.003679</td>
<td>$\xi^{(2)}$</td>
<td>-0.002991</td>
</tr>
<tr>
<td>$A^{(3)}$</td>
<td>-0.000116</td>
<td>$\xi^{(3)}$</td>
<td>0.0000886</td>
</tr>
<tr>
<td>$\alpha_t$</td>
<td>5.747017</td>
<td>$R_e$</td>
<td>3.125850</td>
</tr>
<tr>
<td>$\alpha_q$</td>
<td>4.008572</td>
<td>$R_M$</td>
<td>3.9</td>
</tr>
<tr>
<td>$D^{(0)}$</td>
<td>0.115741</td>
<td>$DQ^{(0)}$</td>
<td>0.133939</td>
</tr>
<tr>
<td>$D^{(1)}$</td>
<td>-0.035967</td>
<td>$DQ^{(1)}$</td>
<td>-0.061012</td>
</tr>
<tr>
<td>$D^{(2)}$</td>
<td>0.00606144</td>
<td>$DQ^{(2)}$</td>
<td>0.0087037</td>
</tr>
<tr>
<td>$D^{(3)}$</td>
<td>-0.00034804</td>
<td>$DQ^{(3)}$</td>
<td>-0.00039698</td>
</tr>
<tr>
<td>$p^{(2)}$</td>
<td>5.706506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p^{(3)}$</td>
<td>12.521950</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p^{(4)}$</td>
<td>13.436660</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p^{(5)}$</td>
<td>6.718205</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p^{(6)}$</td>
<td>1.211088</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.3 Simulation methodology

Gold nanoparticles were modeled as an atomic cluster in which the atoms interact via the force fields given in section 3.2. Monte Carlo simulations on the atomic clusters were carried out using the modified Wang-Landau algorithm described in section 3.1.

The atomic clusters were placed on three-dimensional lattices and an imaginary cubic box was imposed on them so that the atoms of the clusters remain in the box. The clusters were assumed to be in vacuum. The size of the simulation box was set to be twice the size of the global minimum configuration of a given cluster. Periodic boundary conditions were not used because they were not necessary. The finite size effect was
observed at the end of high energy region. Additional simulations were run with larger simulation boxes in that energy region until the box was large enough that no finite size effect was seen. A box size of ten times the global minimum configuration of a cluster was large enough to ensure no finite size effect for all clusters in the study. The Density of States obtained was merged with that for the low energy region and the overlap of the Density of States at the adjoining region was very good.

Starting from a randomly selected initial configuration, a cluster was allowed to change its configuration by two kinds of move: local displacements and a breathing move. In a local displacement move, a randomly selected atom was randomly displaced in the 3-dimensional space within the predetermined magnitude. The breathing move shrank or expanded the whole cluster by a factor of 0.9~1.1. The exploration in the energy space continued until the stop criterion of the algorithm was reached.

In this study, various sizes of atomic clusters were included; they ranged from 10 atoms up to 150 atoms. It was very difficult to study clusters containing more than 150 atoms because of the limitation in the computational power. For all of the employed force fields, the clusters that consist of 10 atoms, 20 atoms, 30 atoms, 40 atoms, 50 atoms, 80 atoms, 100 atoms, and 150 atoms were investigated. In addition, the clusters with 15 atoms, 25 atoms, 35 atoms, 45 atoms, 55 atoms, 60 atoms, 70 atoms, and 90 atoms were studied with the Lennard-Jones potential for a better comparison with literatures. A detailed description of the size of the clusters in this study for each force field is given in Table 3.2.

The radius of gyration ($R_g$) was calculated for all clusters in the study in order to have the sizes of the clusters in length. It can be calculated as
where $N$ is the number of atoms, $\mathbf{r}_i$ is the position of atom $i$ and $\mathbf{r}_{CM}$ is the position of the center of mass of the cluster. The radius of gyration of the global minimum configuration of the clusters as predicted by the various force fields served as the size of the clusters. The diameter of each cluster (twice the radius of gyration) is presented in Table 3.3.

![Table 3.2: Size of the atomic clusters studied with each force field.](image)

<table>
<thead>
<tr>
<th></th>
<th>LJ</th>
<th>LJH</th>
<th>Gupta</th>
<th>SC</th>
<th>SCP</th>
<th>Q-SC</th>
<th>EAM</th>
<th>POT</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>N15</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N20</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>N25</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N30</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>N35</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N40</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>N45</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N50</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>N55</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N60</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N70</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N80</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>N90</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N100</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>N150</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>
Table 3.3: The diameter of gold nanoparticles (twice the radius of gyration) predicted by various force fields. Units are nanometers.

<table>
<thead>
<tr>
<th>N</th>
<th>LJ</th>
<th>Gupta</th>
<th>SC</th>
<th>SCP</th>
<th>QSC</th>
<th>EAM</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.526</td>
<td>0.478</td>
<td>0.48</td>
<td>0.481</td>
<td>0.482</td>
<td>0.49</td>
<td>0.48</td>
</tr>
<tr>
<td>15</td>
<td>0.602</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>0.666</td>
<td>0.634</td>
<td>0.636</td>
<td>0.643</td>
<td>0.628</td>
<td>0.642</td>
<td>0.646</td>
</tr>
<tr>
<td>25</td>
<td>0.716</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>0.768</td>
<td>0.742</td>
<td>0.742</td>
<td>0.75</td>
<td>0.744</td>
<td>0.744</td>
<td>0.78</td>
</tr>
<tr>
<td>35</td>
<td>0.81</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>0.85</td>
<td>0.806</td>
<td>0.808</td>
<td>0.811</td>
<td>0.808</td>
<td>0.818</td>
<td>0.896</td>
</tr>
<tr>
<td>45</td>
<td>0.884</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>50</td>
<td>0.916</td>
<td>0.884</td>
<td>0.882</td>
<td>0.885</td>
<td>0.876</td>
<td>0.892</td>
<td>0.926</td>
</tr>
<tr>
<td>55</td>
<td>0.926</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>60</td>
<td>0.97</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>70</td>
<td>1.02</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>1.068</td>
<td>1.044</td>
<td>1.051</td>
<td>1.058</td>
<td>1.04</td>
<td>1.044</td>
<td>1.078</td>
</tr>
<tr>
<td>90</td>
<td>1.11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>1.152</td>
<td>1.138</td>
<td>1.135</td>
<td>1.157</td>
<td>1.112</td>
<td>1.128</td>
<td>1.164</td>
</tr>
<tr>
<td>150</td>
<td>1.312</td>
<td>1.3</td>
<td>1.303</td>
<td>1.355</td>
<td>1.3</td>
<td>1.298</td>
<td>1.342</td>
</tr>
</tbody>
</table>
The speed of a computer simulation was substantially increased by a technique called “Parallel Simulation”. In the scheme of Parallel Simulation, multiple processors are simultaneously used for the calculations, hence reducing the simulation time by, in principle, the number of the processors. The idea is schematically illustrated in Figure 3.3.

![Diagram of Parallel Simulation](image)

**Figure 3.3**: Schematic illustration of “Parallel Simulation”.

We took advantage of Parallel Simulations by implementing the OpenMP directives into the code. The speed up we achieved by the parallelization of the code was about one-fifth in simulation time compared to that of a serial simulation. The parallelization was performed on an 8-core processor. The source code that was used for the simulation of a 150-atom gold nanoparticle described by the Quantum Sutton-Chen potential (Q-SC) [95] is attached in Appendix to help understand the algorithm and the parallelization.
CHAPTER IV

THERMODYNAMIC PROPERTIES OF GOLD NANOPARTICLES

4.1 Estimation of the Density of States

As discussed in section 2.4, the Density of States of a system can be accurately estimated by a Monte Carlo simulation based on the Wang-Landau algorithm. The Density of States \( g(E) \) was estimated for various sizes of gold nanoparticles (Tables 3.2 and 3.3). A collection of the rescaled logarithm of the Density of States \( g(E) \) is presented in Figure 4.1 for gold nanoparticles described by the Lennard-Jones potential. The figure clearly shows that the number of states increases with increasing energy. It is interesting to note that the Density of States decreases very rapidly as the energy approaches to the global minimum.

The Density of States of gold nanoparticles described by other force fields was estimated as well. The results were similar to that of the Lennard-Jones potential. For this reason, they will not be presented here. The only difference was the absolute value of the energy and the logarithm of the Density of States. The estimated Density of States was used as the starting point for the calculation of the thermodynamic properties and the subsequent analysis of the gold nanoparticle systems.
4.2 Calculation of the thermodynamic properties

Once the Density of States is known for a system, the thermodynamic properties of the system can be calculated using the rules of statistical mechanics. It is an advantage of the Wang-Landau algorithm, as discussed in section 2.4, that the thermodynamic properties may be calculated as a function of temperature from one single simulation.

The most fundamental thermodynamic quantity is the partition function $Q$ which is the sum of the Boltzmann factors. It can be obtained as follows
where \( i \) refers to all microstates and \( E \) refers to the energy levels. Now, various thermodynamic properties may be calculated from the partition function \( Q \). In this study, four thermodynamic quantities, i.e., the free energy \( (F) \), the internal energy \( (U) \), the entropy \( (S) \), and the specific heat \( (C_v) \), over a wide range of temperatures were calculated using the following relations,

\[
Q = \sum_i e^{-\frac{E_i}{k_B T}} = \sum_E g(E) e^{-\frac{E}{k_B T}}
\]

\[
F = -k_B T \ln Q,
\]  

\[
<U> = \frac{\sum_E g(E) \cdot E \cdot e^{-\frac{E}{k_B T}}}{Q},
\]

\[
S = \frac{<U> - F}{T},
\]

\[
C_v = \frac{<U^2> - <U>^2}{k_B T^2}.
\]

The algorithm employed in this study is a modified version of the Wang-Landau algorithm. It is based on the modified Wang-Landau algorithm by Zhou with iterative refining stages which are adopted from the original Wang-Landau algorithm to improve the accuracy of the Density of States estimation as described in section 3.1. Although the algorithm consists of the features of the two well-established algorithms, its validity must be confirmed separately because the mixing of the features of successive algorithms does
not ensure the success of the resultant algorithm. The specific heat ($C_v$) of gold nanoparticles described by the Lennard-Jones potential was calculated and compared with the values from other simulation [59]. Figure 4.2 shows that the algorithm successfully reproduces the specific heat of the Lennard-Jones clusters. Having validated the algorithm, the simulation of gold nanoparticles was extended to include a variety of force fields.

The calculated thermodynamic properties of gold nanoparticles described by the Lennard-Jones potential are presented as a function of temperature. Figure 4.3 shows the free energy ($F$), Figure 4.4 shows the internal energy ($U$), Figure 4.5 shows the entropy ($S$), and Figure 4.6 shows the specific heat ($C_v$). They are normalized for a better view. It is a well-known fact that an abnormal behavior of a thermodynamic property is an indication of a structural transition, such as melting. The structural transition of the clusters is evidenced in the internal energy, the entropy, and most clearly in the specific heat plots in Figures 4.4 to 4.6. For instance, the 150-atom cluster shows a jump in the internal energy and the entropy, and a peak in the specific heat, at the temperature of 230 K. A comparison with literature data [59] showed that the structural transition associated with the peak in Figure 4.6 is the melting of the whole cluster.
Figure 4.2: Specific heat ($C_v$) of Lennard-Jones clusters as a function of temperature. The upper plot shows the results of this study while the lower figures show the results of other Monte Carlo simulation [59]. The numbers on the y-axis represent the number of atoms in the clusters. (Reprinted with permission from [59]. Copyright 2006 American Institute of Physics)
Figure 4.3: Calculated free energy ($F$) of gold nanoparticles described by the Lennard-Jones potential (LJ) as a function of temperature.

Figure 4.4: Calculated internal energy ($U$) of gold nanoparticles described by the Lennard-Jones potential (LJ) as a function of temperature.
Figure 4.5: Calculated entropy ($S$) of gold nanoparticles described by the Lennard-Jones potential (LJ) as a function of temperature.

Figure 4.6: Calculated specific heat ($C_v$) of gold nanoparticles described by the Lennard-Jones potential (LJ) as a function of temperature.
The fluctuations in the curve in the low-temperature region in Figure 4.6 are due to numerical inaccuracies and do not carry any physical meaning. Two methods are commonly employed to remove these fluctuations and improve the description of the low temperature behavior. The first one is the use of Quantum Mechanics and the other one is the use of an advanced algorithm which can effectively sample the low energy region. In fact, they have already been applied to a couple of systems where the computational load is not so large [59, 60].

As mentioned in section 1.6, one of the main objectives of this dissertation is the testing of the accuracy of the various force fields. In order to do that, the gold nanoparticles are modeled by the carefully selected eight force fields discussed in section 3.2. The Density of States of gold nanoparticles estimated by these force fields showed qualitatively similar behavior. The thermodynamic properties, however, differed significantly from one force field to the other and this serves as the starting point for the evaluation of the force fields.

The Lennard-Jones potential with Heinz’s parameterization (LJH) was employed in this study in the hope that it would provide a better description of gold nanoparticles than the original Lennard-Jones potential. However, despite the use of a different parameterization, it was just as same as the original Lennard-Jones potential in estimating the melting temperature of gold nanoparticles as the specific heat \( (C_v) \) calculated from the two parameterizations showed no differences (see Figure 4.7).
Figure 4.7: Specific heat ($C_v$) of gold nanoparticles described by the Lennard-Jones potentials. Solid line: original parameterization, (○): Heinz’s parameterization.

Figures 4.8 to 4.31 show the normalized, calculated thermodynamic properties of gold nanoparticles described by the various force fields employed in this study. In general, the melting of the cluster occurs over a broad range of temperatures in small clusters (clusters of 10-, 20-, 30-atoms). Because of this, defining a finite melting temperature for the small clusters was not straightforward. For large clusters (clusters of 80-, 100-, 150-atoms), the melting point occurs in a relatively narrow range of temperatures and the prediction of the melting temperature could be done quickly.
Figure 4.8: Calculated free energy ($F$) of gold nanoparticles described by the Gupta potential as a function of temperature.

Figure 4.9: Calculated internal energy ($U$) of gold nanoparticles described by the Gupta potential as a function of temperature.
Figure 4.10: Calculated entropy ($S$) of gold nanoparticles described by the Gupta potential as a function of temperature.

Figure 4.11: Calculated specific heat ($C_v$) of gold nanoparticles described by the Gupta potential as a function of temperature.
Figure 4.12: Calculated free energy \( (F) \) of gold nanoparticles described by the Sutton-Chen potential (SC) as a function of temperature.

Figure 4.13: Calculated internal energy \( (U) \) of gold nanoparticles described by the Sutton-Chen potential (SC) as a function of temperature.
Figure 4.14: Calculated entropy ($S$) of gold nanoparticles described by the Sutton-Chen potential (SC) as a function of temperature.

Figure 4.15: Calculated specific heat ($C_v$) of gold nanoparticles described by the Sutton-Chen potential (SC) as a function of temperature.
Figure 4.16: Calculated free energy ($F$) of gold nanoparticles described by the Sutton-Chen potential with Pawluk’s parameterization (SCP) as a function of temperature.

Figure 4.17: Calculated internal energy ($U$) of gold nanoparticles described by the Sutton-Chen potential with Pawluk’s parameterization (SCP) as a function of temperature.
Figure 4.18: Calculated entropy ($S$) of gold nanoparticles described by the Sutton-Chen potential with Pawluk’s parameterization (SCP) as a function of temperature.

Figure 4.19: Calculated specific heat ($C_v$) of gold nanoparticles described by the Sutton-Chen potential with Pawluk’s parameterization (SCP) as a function of temperature.
Figure 4.20: Calculated free energy ($F$) of gold nanoparticles described by the Quantum Sutton-Chen potential (Q-SC) as a function of temperature.

Figure 4.21: Calculated internal energy ($U$) of gold nanoparticles described by the Quantum Sutton-Chen potential (Q-SC) as a function of temperature.
Figure 4.22: Calculated entropy ($S$) of gold nanoparticles described by the Quantum Sutton-Chen potential (Q-SC) as a function of temperature.

Figure 4.23: Calculated specific heat ($C_v$) of gold nanoparticles described by the Quantum Sutton-Chen potential (Q-SC) as a function of temperature.
Figure 4.24: Calculated free energy ($F$) of gold nanoparticles described by the Embedded Atom Method (EAM) by Cai and Ye as a function of temperature.

Figure 4.25: Calculated internal energy ($U$) of gold nanoparticles described by the Embedded Atom Method (EAM) by Cai and Ye as a function of temperature.
Figure 4.26: Calculated entropy ($S$) of gold nanoparticles described by the Embedded Atom Method (EAM) by Cai and Ye as a function of temperature.

Figure 4.27: Calculated specific heat ($C_v$) of gold nanoparticles described by the Embedded Atom Method (EAM) by Cai and Ye as a function of temperature.
Figure 4.28: Calculated free energy ($F$) of gold nanoparticles described by the empirical potential for gold proposed by Olivier (POT) as a function of temperature.

Figure 4.29: Calculated internal energy ($U$) of gold nanoparticles described by the empirical potential for gold proposed by Olivier (POT) as a function of temperature.
Figure 4.30: Calculated entropy ($S$) of gold nanoparticles described by the empirical potential for gold proposed by Olivier (POT) as a function of temperature.

Figure 4.31: Calculated specific heat ($C_v$) of gold nanoparticles described by the empirical potential for gold proposed by Olivier (POT) as a function of temperature.
The thermodynamic information at low temperatures was discarded for all employed force fields. In the case of Lennard-Jones potential, there were fluctuations in the low temperature region of the specific heat. A correct description of the low temperature behavior was not achieved with any of the tested force fields. This is not only because of the use of Classical Physics, but also due to the absence of an algorithm which effectively samples the low energy region. However, the computational cost for the use of quantum mechanics in studies of complex systems, such as the thermodynamic study of metal nanoparticles, is so large that only very small systems (tens of atoms) have been investigated with quantum mechanics [133].

As shown in Figures 4.8 to 4.31, the melting point of gold nanoparticles is size-dependent for the studied size ranges. This is in accordance with many experimental observations and simulation results [68-73]. An interesting observation is that the melting point predicted for a cluster differs by the force field. In the next section, the accuracy of the employed force fields in predicting the melting point of gold nanoparticles is evaluated by a thorough comparison with theory and experiments.

4.3 Melting of gold nanoparticles

The melting point of gold nanoparticles as predicted by various force fields is presented in Table 4.1. The prediction of the melting point was not possible for some of the small clusters because their thermodynamic properties did not show a clear sign of melting.
Table 4.1: Melting point of gold nanoparticles as predicted by the various force fields and by the extrapolation of experimental data [69, 75]. N is the number of atoms and D is the diameter. * represents the clusters whose melting point could not be identified. The melting points are in Kelvin.

<table>
<thead>
<tr>
<th>N</th>
<th>Lennard-Jones</th>
<th></th>
<th>Gupta</th>
<th></th>
<th>Sutton-Chen</th>
<th></th>
<th>SCP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D (nm)</td>
<td>T&lt;sub&gt;m&lt;/sub&gt;</td>
<td>D (nm)</td>
<td>T&lt;sub&gt;m&lt;/sub&gt;</td>
<td>D (nm)</td>
<td>T&lt;sub&gt;m&lt;/sub&gt;</td>
<td>D (nm)</td>
<td>T&lt;sub&gt;m&lt;/sub&gt;</td>
</tr>
<tr>
<td>10</td>
<td>0.53</td>
<td>*</td>
<td>0.48</td>
<td>*</td>
<td>0.48</td>
<td>*</td>
<td>0.48</td>
<td>120</td>
</tr>
<tr>
<td>20</td>
<td>0.67</td>
<td>*</td>
<td>0.63</td>
<td>310</td>
<td>0.64</td>
<td>*</td>
<td>0.64</td>
<td>140</td>
</tr>
<tr>
<td>30</td>
<td>0.77</td>
<td>*</td>
<td>0.74</td>
<td>330</td>
<td>0.74</td>
<td>290</td>
<td>0.75</td>
<td>170</td>
</tr>
<tr>
<td>40</td>
<td>0.85</td>
<td>128</td>
<td>0.81</td>
<td>410</td>
<td>0.81</td>
<td>380</td>
<td>0.81</td>
<td>190</td>
</tr>
<tr>
<td>50</td>
<td>0.92</td>
<td>163</td>
<td>0.88</td>
<td>400</td>
<td>0.88</td>
<td>380</td>
<td>0.89</td>
<td>190</td>
</tr>
<tr>
<td>60</td>
<td>0.97</td>
<td>182</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>70</td>
<td>1.02</td>
<td>190</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>1.07</td>
<td>200</td>
<td>1.04</td>
<td>440</td>
<td>1.05</td>
<td>400</td>
<td>1.06</td>
<td>200</td>
</tr>
<tr>
<td>90</td>
<td>1.11</td>
<td>205</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>1.15</td>
<td>211</td>
<td>1.14</td>
<td>450</td>
<td>1.14</td>
<td>430</td>
<td>1.16</td>
<td>210</td>
</tr>
<tr>
<td>150</td>
<td>1.31</td>
<td>229</td>
<td>1.30</td>
<td>540</td>
<td>1.30</td>
<td>520</td>
<td>1.36</td>
<td>320</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N</th>
<th>Q-SC</th>
<th></th>
<th>EAM by Cai</th>
<th></th>
<th>POT</th>
<th></th>
<th>Experiments</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D (nm)</td>
<td>T&lt;sub&gt;m&lt;/sub&gt;</td>
<td>D (nm)</td>
<td>T&lt;sub&gt;m&lt;/sub&gt;</td>
<td>D (nm)</td>
<td>T&lt;sub&gt;m&lt;/sub&gt;</td>
<td>D (nm)</td>
<td>T&lt;sub&gt;m&lt;/sub&gt;</td>
</tr>
<tr>
<td>10</td>
<td>0.48</td>
<td>*</td>
<td>0.49</td>
<td>*</td>
<td>0.48</td>
<td>*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>0.63</td>
<td>*</td>
<td>0.64</td>
<td>*</td>
<td>0.65</td>
<td>*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>0.74</td>
<td>300</td>
<td>0.74</td>
<td>*</td>
<td>0.78</td>
<td>1010</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>0.81</td>
<td>280</td>
<td>0.82</td>
<td>290</td>
<td>0.90</td>
<td>*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>50</td>
<td>0.88</td>
<td>500</td>
<td>0.89</td>
<td>310</td>
<td>0.93</td>
<td>1090</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>60</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>70</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>1.04</td>
<td>550</td>
<td>1.04</td>
<td>350</td>
<td>1.08</td>
<td>1135</td>
<td>1.00</td>
<td>64</td>
</tr>
<tr>
<td>90</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.10</td>
<td>179</td>
</tr>
<tr>
<td>100</td>
<td>1.11</td>
<td>580</td>
<td>1.13</td>
<td>360</td>
<td>1.16</td>
<td>1150</td>
<td>1.20</td>
<td>275</td>
</tr>
<tr>
<td>150</td>
<td>1.30</td>
<td>680</td>
<td>1.30</td>
<td>470</td>
<td>1.34</td>
<td>1170</td>
<td>1.30</td>
<td>357</td>
</tr>
</tbody>
</table>
The estimated melting temperature of gold nanoparticles shown in Table 4.1 was compared with theory [69], experimental observations [69, 75], and other simulation results [71, 72] to evaluate the accuracy of each force field in describing the thermal behavior of gold nanoparticles and is shown in Figure 4.32.

Figure 4.32: Melting point of gold nanoparticles. Solid line: theory (the modified Pawlow relation (equation (1.2))) [69], black squares: experimental [69, 75], dotted line: the least square fit of the experimental data, pink diamonds: MD simulation with MEAM [71], blue diamonds: MD simulation with EAM [72]. Circles: prediction of current study (empty: prediction with the LJ potential, skyblue: prediction with the Gupta potential, navy: prediction with the SC potential, orange: prediction with the SCP potential, red: prediction with the Q-SC potential, violet: prediction with the EAM by Cai and Ye potential, green: prediction with the POT potential.
The experimental data were measured using a scanning electron-diffraction technique. Pure gold nanoparticles were formed by the condensation of gold vapor on a thin amorphous carbon film and were placed in a vacuum furnace with sample holder made from graphite with spectroscopic purity which is known to have low affinity for gold. The heating of the furnace was done by electron bombardment.

The least square fit of the experimental data was created in order to better compare the simulation results with the experimental values. Since no experimental data were available in the studied size range, the predicted melting point was compared to the extrapolated values of the experimental data. The least square fit of the experimental data was obtained by using the functional form of \( T_m(d) = A - B/d \) because it had been commonly acknowledged in theoretical studies of melting of metal nanoparticles that the depression in the melting point is inversely proportional to the size of the nanoparticles. The modified Pawlow relation (equation (1.2)) [69],

\[
\frac{T_m(d)}{T_m(\infty)} = 1 - \frac{4}{\rho_s L} \left( \gamma_s - \gamma_l \left( \frac{\rho_s}{\rho_l} \right)^{2/3} \right) \frac{1}{d}
\]

(1.2)

is a kind of such. For gold nanoparticles, the modified Pawlow relation can be simplified into

\[
T_m(d) = 1337 - \frac{911}{d}
\]

(4.6)

with the diameter in the unit of nanometer, using the experimental values of \( \rho_s = 18400 \) kg/m\(^3\) [75], \( \rho_l = 17280 \) kg/m\(^3\) [75], \( \gamma_s = 1.38 \) J/m\(^2\) [134], \( \gamma_l = 1.135 \) J/m\(^2\) [75], \( L = 6.27 \times 10^4 \).
J/kg [75], and \( T_{m(\infty)} = 1337 \) K [71]. It is plotted as a solid line in Figure 4.32. The extrapolation curve of the experimental data was obtained to be

\[
T_{m}(d) = 1335 - \frac{1272}{d}
\]  

(4.7)

with the diameter in the unit of nanometer and is plotted as a dotted line in Figure 4.32.

The melting point of gold nanoparticles estimated by the Molecular Dynamics simulation using the Modified Embedded Atom Model (MEAM) [71] showed an excellent agreement with the experimental data for the nanoparticles with size larger than 1.6 nanometers while that using the Embedded Atom Method (EAM) [72] was underestimated. However, the accuracy of the MEAM potential remains unknown for gold nanoparticles with size smaller than 1.6 nm.

It is clear in Figure 4.32 that the melting point predicted by the empirical potential for gold by Olivier and coworkers (POT) does not conform to the theory or the extrapolated values of the experimental observations in the studied size range. The melting point predicted by the Lennard-Jones potential (LJ), the Gupta potential, the Sutton-Chen potential (SC), the Sutton-Chen potential with Pawluk’s parameterization (SCP), the Quantum Sutton-Chen potential (Q-SC), and the Embedded Atom Model (EAM) by Cai and Ye is near the prediction of the modified Pawlow relation and/or the extrapolated value of the experimental data in the studied size range.

The extrapolated region of the least square fit of the experimental data may contain a large error since there is no data point in that region. Thus, it is not a dependable criterion to tell whether the simulation results are accurate or not in this study. A least square fit using the experimental data and the simulation data could serve as a
criterion to determine the accuracy of the simulation results since it includes the simulation data in the fitting procedure.

The accuracy of the employed force fields in predicting the melting point of gold nanoparticles with size smaller than 1.3 nm was evaluated by calculating the standard deviation of the simulation results from the least square fit that was created from the experimental and the simulation data. In addition, the slope of the linear regression fit of the simulation results was compared to that of the least square fit at the intersection of the two fitting functions. For the Quantum Sutton-Chen potential (Q-SC), the slope of the two functions was compared at the median of the studied size because there was no intersection. The least square fit using the experimental and the simulation data along with the linear regression fit of the simulation data are shown for each force field in Figure 4.33 to Figure 4.38.
Figure 4.33: Melting point of gold nanoparticles predicted by the Lennard-Jones potential (LJ) compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Lennard-Jones potential, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data.
Figure 4.3: Melting point of gold nanoparticles predicted by the Gupta potential compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Gupta potential, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data.
Figure 4.35: Melting point of gold nanoparticles predicted by the Sutton-Chen potential (SC) compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Sutton-Chen potential, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data.
Figure 4.36: Melting point of gold nanoparticles predicted by the Sutton-Chen potential with Pawluk’s parameterization for small clusters (SCP) compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Sutton-Chen potential with Pawluk’s parameterization, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data.
Figure 4.37: Melting point of gold nanoparticles predicted by the Quantum Sutton-Chen potential (Q-SC) compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Quantum Sutton-Chen potential, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data.
Figure 4.38: Melting point of gold nanoparticles predicted by the Embedded Atom Model (EAM) by Cai and Ye compared with a theory [69] and experimental data [69, 75]. Squares: experimental, circles: simulation with the Embedded Atom Model (EAM) by Cai and Ye, solid line: theory (the modified Pawlow relation [69]), dotted line: the least square fit of the experimental data, dashed line: the least square fit using experimental and simulation data. The straight line that goes through the circles is the linear regression fit of the simulation data.
The standard deviation of the predicted melting points from the least square fit is shown in Table 4.2 along with that of the experimental data. Among the force fields employed in this study, the Quantum Sutton-Chen potential (Q-SC) had the smallest standard deviation of 89.6 K. It is larger than that of the experimental data (31.4 K) but is comparable to that. The Embedded Atom Model (EAM) by Cai and Ye, the Lennard-Jones potential (LJ), the Sutton-Chen potential (SC), and the Gupta potential had a little higher value for the standard deviation compared to that of the Quantum Sutton-Chen potential (Q-SC). The Sutton-Chen potential with Pawluk’s parameterization (SCP) had the standard deviation of 281.2 K which is about nine times of that of the experimental data. The inaccurate description of the melting point of gold nanoparticles by the Sutton-Chen potential with Pawluk’s parameterization (SCP) is clearly seen in Figure 4.36.

Table 4.2: Standard deviation of the predicted melting points from the least square fit of them with the experimental data.

<table>
<thead>
<tr>
<th>Force Field</th>
<th>Standard deviation (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LJ</td>
<td>112.4</td>
</tr>
<tr>
<td>Gupta</td>
<td>135.3</td>
</tr>
<tr>
<td>SC</td>
<td>120.7</td>
</tr>
<tr>
<td>SCP</td>
<td>281.2</td>
</tr>
<tr>
<td>Q-SC</td>
<td>89.6</td>
</tr>
<tr>
<td>EAM by Cai and Ye</td>
<td>111.0</td>
</tr>
<tr>
<td>Experiments</td>
<td>31.4</td>
</tr>
</tbody>
</table>
The slope of the linear regression fit of the predicted melting points and of the least square fit at the intersection with the linear regression fit (at the median of the studied size for the Quantum Sutton-Chen potential (Q-SC)) is shown in Table 4.3. The discrepancy between them is shown together. This could serve as an additional criterion to tell the accuracy of the force fields in predicting the melting point of gold nanoparticles in the studied size range. It is interesting to note that the difference between the two slopes is only 3% in the case of the Quantum Sutton-Chen potential (Q-SC). The agreement of the two slopes was not good in all the other force fields.

Table 4.3: The slope of the linear regression fit of the predicted melting points and of the least square fit at the intersection with the linear regression fit. For the Quantum Sutton-Chen potential (Q-SC), the slope of the least square fit was obtained at the median of the studied sizes.

<table>
<thead>
<tr>
<th>Force Field</th>
<th>Slope of the linear regression</th>
<th>Slope at the intersection</th>
<th>% error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LJ</td>
<td>205</td>
<td>1090</td>
<td>81%</td>
</tr>
<tr>
<td>Gupta</td>
<td>314</td>
<td>900</td>
<td>65%</td>
</tr>
<tr>
<td>SC</td>
<td>327</td>
<td>828</td>
<td>61%</td>
</tr>
<tr>
<td>SCP</td>
<td>194</td>
<td>1437</td>
<td>87%</td>
</tr>
<tr>
<td>Q-SC</td>
<td>713</td>
<td>738</td>
<td>3%</td>
</tr>
<tr>
<td>EAM by Cai and Ye</td>
<td>351</td>
<td>845</td>
<td>58%</td>
</tr>
</tbody>
</table>
From the comparison of the predicted melting point of gold nanoparticles with the experimental observations as discussed above, it can be said that the Quantum Sutton-Chen potential (Q-SC) accurately describes the melting point of gold nanoparticles with size smaller than 1.3 nm. The success of the Quantum Sutton-Chen potential (Q-SC) is attributed to its quantum corrections that are added to the original Sutton-Chen potential. The Quantum Sutton-Chen potential (Q-SC) could serve as an adequate alternative to the quantum mechanical calculations with much less computational cost.

The Lennard-Jones potential (LJ), the Gupta potential, the Sutton-Chen potential (SC), the Sutton-Chen potential with Pawluk’s parameterization (SCP), the Embedded Atom Method (EAM) by Cai and Ye, and the empirical potential for gold by Olivier and coworkers (POT) did not describe the melting behavior of gold nanoparticles correctly in the studied size range. The underestimation of the melting point of the Lennard-Jones potential (LJ) is attributed to the absence of a term that describes the metallic cohesion. The failure of the Gupta potential, the Sutton-Chen potential (SC), and the Embedded Atom Method (EAM) by Cai and Ye could be attributed to the fact that these potentials were developed to fit the bulk properties, thus showing some deviations for small nanoparticles. The failure of the Sutton-Chen potential with Pawluk’s parameterization (SCP) might be due to the fact that the parameterization had been developed for extremely small systems such as dimers and trimmers. It can be said for sure that the parameterization is not valid for larger systems. The failure of the empirical potential for gold proposed by Olivier and coworkers (POT) in spite of its angular corrections could be attributed to its development by considering only the energetics of the model systems.
4.4 Conclusions

In this Chapter, the thermodynamic properties of gold nanoparticles with sizes smaller than 1.3 nm were investigated using various force fields for gold and the accuracy of the employed force fields were evaluated by analyzing the estimated melting point. The size-dependence of the melting point was qualitatively well predicted by every force field employed. However, a thorough quantitative comparison of the predicted melting point with the extrapolation of the experimental observations revealed that most of the employed force fields (the Lennard-Jones potential (LJ), the Gupta potential, the Sutton-Chen potential (SC), the Sutton-Chen potential with Pawluk’s parameterization (SCP), the Embedded Atom model (EAM) by Cai and Ye, and the empirical potential for gold proposed by Olivier and coworkers (POT)) failed to provide a quantitatively correct description of the melting behavior of gold nanoparticles with size smaller than 1.3 nm. Meanwhile, the Quantum Sutton-Chen potential (Q-SC) successfully described the melting behavior of gold nanoparticles with sizes smaller than 1.3 nm, implying that it may be used for extended studies. They can be used to study other physical properties of gold nanoparticles such as mechanical properties, larger sizes of gold nanoparticles, or even different kind of metals. In fact, the parameterizations for many face-centered cubic (fcc) metals are already available for the Quantum Sutton-Chen potential (Q-SC).
CHAPTER V

CONCLUSIONS

The thermodynamic properties of gold nanoparticles were investigated by Monte Carlo simulations with various force fields. Monte Carlo simulations with the modified Wang-Landau algorithm successfully estimated the Density of States and subsequently, the thermodynamic properties, such as size-dependent melting point, of gold nanoparticles with sizes smaller than 1.3 nm. Based on the validation and the effectiveness of the method on gold nanoparticle systems showed in this work, larger nanoparticles will be studied with the current method as more computational power becomes available.

The thermodynamic behavior of gold nanoparticles at low temperatures can be described with an advanced algorithm which effectively explores the low energy region of the potential energy space.

Among the many currently available force fields for metals, the Quantum Sutton-Chen potential (Q-SC) provided a correct description of the melting behavior of gold nanoparticles with sizes smaller than 1.3 nm. The Lennard-Jones potential (LJ), the Gupta potential, the Sutton-Chen potential (SC), the Sutton-Chen potential with Pawluk’s parameterization (SCP), the Embedded Atom Method (EAM) by Cai and Ye, and the empirical potential for gold proposed by Olivier and coworkers (POT) were able to
provide a qualitatively good description of the melting behavior of gold nanoparticles with sizes smaller than 1.3 nm. However, no quantitatively correct description of the melting behavior could be achieved with these force fields.

The success of the Quantum Sutton-Chen potential (Q-SC) is attributed to its quantum corrections. The Quantum Sutton-Chen potential (Q-SC) could serve as a great alternative to quantum mechanical calculations for the studies of large metal nanoparticles since they can provide accurate description for them with far less computational cost.
BIBLIOGRAPHY


/* This code reads coordinates of N atoms, converts them into integers, and evaluates the Quantum Sutton-Chen potential, then performs Wang-Landau algorithm with global update proposed by Dr. Zhou in 1-D energy space until the modification factor becomes less than 1.000 000 01. */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <stdbool.h>
#include <omp.h>
#include "/opt/intel/cmkl/10.0.3.020/include/mkl.h"
#include "/home/kingrice/dSFMT-src-2.0/dSFMT.h"
#include "/home/kingrice/dSFMT-src-2.0/dSFMT-params.h"

#define ATOMNO 150      // number of atoms of the cluster
#define LATTICECONSTANT 4.0651      // lattice constant "a"
#define OMEGA 0.1       // the value of DOS above which we apply global update
#define KAPPA 10.0      // DOS increment by a global update
#define LAMBDA 10000.0  // a parameter used in global update
#define PI 3.141592654  // the constant Pi

// Declaration of pointers to files
FILE *filecoorpointer;       // call the file that carries the coordinates of the initial configuration
FILE *filecrystal;            // file that contains energy value and coordinates of lowest energy state found
FILE *filedos;                // file that contains density of states

// Declaration of variables, arrays, and structures
const short g_csatomnoplus = ATOMNO+1;     // auxiliary constant for speed up
struct latticestruct
struct latticestruct g_ilatticeatom[g_csatomnoplus]; // coordinates of atoms (m,n,p)
struct latticestruct g_ilatticeatom_aux[g_csatomnoplus]; // coordinates of atoms - auxiliary
int g_idist_atom[g_csatomnoplus][g_csatomnoplus]; // distance square between two atoms
int g_idist_atom_aux[g_csatomnoplus][g_csatomnoplus]; // used in local moves

// Variables for random number generation
/** dsfmt internal state vector */
dsfmt_t dsfmt;
dsfmt_t dsfmt_global_data;
/** dsfmt mexp for check */
static const int dsfmt_mexp = DSFMT_MEXP;

// FUNCTIONS BEGIN HERE ---------------------------
// Initialization of the arrays to zero. CHECKED OK
void Initialize_arrays (double (*V)[g_csatomnoplus], double (*V_temp)[g_csatomnoplus], double (*ratio8)[g_csatomnoplus], double (*ratio8_temp)[g_csatomnoplus], double *rho, double *rho_temp, int nofbin,unsigned long long int *histogram,double *lnge,double *lnge2,double *lnge3)
{
    register short a1, b1;
    register short a2, b2;
    a1=ATOMNO;
    for(;a1--;)
    {
        a2=a1+1;
        b1=a2;
        for(;b1--;
            {
                b2=b1+1;
                g_idist_atom[a2][b2]=0;
                g_idist_atom[b2][a2]=0;
                g_idist_atom_aux[a2][b2]=0;
                g_idist_atom_aux[b2][a2]=0;
                V[a2][b2]=0.0;
                V[b2][a2]=0.0;
                V_temp[a2][b2]=0.0;
                V_temp[b2][a2]=0.0;
                ratio8[a2][b2]=0.0;
                ratio8[b2][a2]=0.0;
            }
        }
    }
}
ratio8_temp[a2][b2]=0.0;
ratio8_temp[b2][a2]=0.0;
}
rho[a2]=0.0;
rho_temp[a2]=0.0;
}
b1=nofbin;
for(;b1--;)
{
    histogram[b1]=0.0;
    Inge[b1]=0.0;
    Inge2[b1]=0.0;
    Inge3[b1]=0.0;
}
}

// Conversion of real coordinates to lattice coordinates. CHECKED OK
void Latticecoordinate (int nu, float x, float y, float z)
{
    const float cfinverse_a = 100.0/LATTICECONSTANT;     // variable used in the
Latticecoordinate function for speed up
    g_iatticeatom[nu].x=floor((x*cfinverse_a)+0.5);
    g_iatticeatom[nu].y=floor((y*cfinverse_a)+0.5);
    g_iatticeatom[nu].z=floor((z*cfinverse_a)+0.5);
}

// Reading initial coordinates from a pdb file from a dat file. CHECKED OK
void Readinitialcoordinates ()
{
    register short a1;
    char trashinfo[100];     // variable to carry unnecessary lines in the pdb file
    char terminal[4] = "";    // used to find the end-of-file signal in the pdb file
    short no;                // variable to carry ATOM number from in the pdb file
    char str[16];            // variable to carry unnecessary characters in the pdb file
    short check_atomno = 0;  // used to confirm that atomno is the correct number of
atoms
    _Bool signal = 1;        // signal used as a boolean variable for do - while loop
    float l_fx, l_fy, l_fz;  // load coordinates onto these variables

    if
((filecoorpointer=fopen("/home/kingrice/bin/lib/coordinate150sorted.pdb","r"))==NULL)
{
    system("clear");
    printf("WARNING !!! \nError reading initial coordinate file \nPlease, check\n");
    exit(1);
}
// Reading coordinates from pdb file. CHECKED OK
a1=5;
for(;a1--;)  // Skip the first five lines of coorpointer. Note the reverse loop for speed up
    fgets(trashinfo,100,filecoorpointer);
do
    {
        fscanf(filecoorpointer,"%s",terminal);
        if(strcmp(terminal,"TER"))
        {
            fscanf(filecoorpointer,"%d %s %s %s %f %f %f %s %s %s %s
","no,str,str,str,&l_fx,&l_fy,&l_fz,str,str,str,str);
            Latticecoordinate(no,l_fx,l_fy,l_fz);
            check_atomno++;
        }
        else signal=0;
    }
while(signal);

// Checking if the number of atoms read from the pdb file is equal to the ATOMNO set
// at the beginning of this code
if(check_atomno!=ATOMNO)
{
    system("clear");
    printf("WARNING !!! 
The number of atoms in the file does not agree with the
number entered in atomno. 
Please, check\n");
    exit(1);
}

// Determination of the size of the simulation box from the coordinates. CHECKED OK
int Setboxsize ()
{
    register short a1, b1;
    int maxc;  // maximum of coordinates
    int x,y,z;  // speed up variables
    maxc=0;
a1=ATOMNO;
for(;a1--;)
{
    b1=a1+1;
    x=abs(g_ilatticeatom[b1].x);
    y=abs(g_ilatticeatom[b1].y);
    z=abs(g_ilatticeatom[b1].z);
    if(1+((x-maxc)>>31))
maxc=x;
if(1+((y-maxc)>>31))
maxc=y;
if(1+((z-maxc)>>31))
maxc=z;
}
// Determining the size of the simulation box. CHECKED OK
maxc=2*maxc;
printf("box size=%d\n",maxc);
return (maxc);
}

// Computation of the initial distances between atoms. CHECKED OK
void Distances ()
{
    register short a1, b1;
    register short a2, b2;
    int im, in, ip, idm, idn, idp;
    a1=ATOMNO-1;
    for(;a1--;)
    {
        a2=a1+2;
        im=g_ilatticeatom[a2].x;
        in=g_ilatticeatom[a2].y;
        ip=g_ilatticeatom[a2].z;
        b1=a2-1;
        for(;b1--;)
        {
            b2=b1+1;
            idm=g_ilatticeatom[b2].x-im;
            idn=g_ilatticeatom[b2].y-in;
            idp=g_ilatticeatom[b2].z-ip;
            g_idist_atom[a2][b2]=(idm*idm)+(idn*idn)+(idp*idp);
            g_idist_atom[b2][a2]=g_idist_atom[a2][b2];
        }
    }
}

// Computation of the initial energy and filling V[], ratio8[], and rho[]. CHECKED OK
double EnergyInitial (double (*V)[g_csatomnoplus], double (*ratio8)[g_csatomnoplus],
double *rho, double c)
{
    register short a1, b1;
    register short a2, b2;
    double repulsive, attractive, ratio2, ratio3, totalenergy;

repulsive=0.0;
attractive=0.0;

a2=ATOMNO-1;
for(;a2--;
{
    a1=a2+2;
b2=a2+1;
for(;b2--;)
    {
        b1=b2+1;
        ratio2=10000.0/g_idist_atom[a1][b1];
        ratio3=ratio2*sqrt(ratio2);
        //printf("%f ",ratio2);
        // attractive part - filling rho components (a/r_ij)^8 matrix
        ratio8[a1][b1]=ratio3*ratio3*ratio2;
        ratio8[b1][a1]=ratio8[a1][b1];
        // repulsive part - filling (a/r_ij)^10(=V(r_ij)) matrix
        V[b1][a1]=ratio3*ratio8[b1][a1];
        V[a1][b1]=V[b1][a1];
        // repulsive energy calculation
        repulsive+=V[a1][b1];
    }
}
//printf("repulsive energy = %lf\n",repulsive);

// filling rho array (n-body contribution)
a2=ATOMNO;
for(;a2--;
{
    a1=a2+1;
rho[a1]=0.0;
b2=ATOMNO;
for(;b2--;
    {
        b1=b2+1;
        rho[a1]+=ratio8[a1][b1]; // make sure ratio8[i][i]=0.0
    }
    // attractive energy calculation
    attractive+=sqrt(rho[a1]);
}
//printf("attractive energy = %lf\n",g_dc*attractive);
// calculation of total energy
totalenergy = repulsive - c*attractive;
return(totalenergy);
}

/*-----------------*
 STATIC FUNCTIONS
 ----------------*/
inline static uint32_t ini_func1(uint32_t x);
inline static uint32_t ini_func2(uint32_t x);
inline static void gen_rand_array_c1o2(dsfmt_t *dsfmt, w128_t array[],
    int size);
inline static void gen_rand_array_c0o1(dsfmt_t *dsfmt, w128_t array[],
    int size);
inline static void gen_rand_array_o0c1(dsfmt_t *dsfmt, w128_t array[],
    int size);
inline static void gen_rand_array_o0o1(dsfmt_t *dsfmt, w128_t array[],
    int size);
inline static int idxof(int i);
static void initial_mask(dsfmt_t *dsfmt);
static void period_certification(dsfmt_t *dsfmt);

#if defined(HAVE_SSE2)
#  include <emmintrin.h>
/** mask data for sse2 */
static __m128i sse2_param_mask;
/** 1 in 64bit for sse2 */
static __m128i sse2_int_one;
/** 2.0 double for sse2 */
static __m128d sse2_double_two;
/** -1.0 double for sse2 */
static __m128d sse2_double_m_one;

static void setup_const(void);
#endif

/**
 * This function simulate a 32-bit array index overlapped to 64-bit
 * array of LITTLE ENDIAN in BIG ENDIAN machine.
 */
#if defined(DSFMT_BIG_ENDIAN)
inline static int idxof(int i) {
    return i ^ 1;
}
#else
inline static int idxof(int i) {
    return i;
}
/**
 * This function represents the recursion formula.
 * @param r output
 * @param a a 128-bit part of the internal state array
 * @param b a 128-bit part of the internal state array
 * @param lung a 128-bit part of the internal state array
 */

#if defined(HAVE_ALTIVEC)
inline static void do_recursion(w128_t *r, w128_t *a, w128_t *b, w128_t *lung) {
const vector unsigned char sl1 = ALTI_SL1;
const vector unsigned char sl1_perm = ALTI_SL1_PERM;
const vector unsigned int sl1_msk = ALTI_SL1_MSK;
const vector unsigned char sr1 = ALTI_SR;
const vector unsigned char sr1_perm = ALTI_SR_PERM;
const vector unsigned int sr1_msk = ALTI_SR_MSK;
const vector unsigned char perm = ALTI_PERM;
const vector unsigned int msk1 = ALTI_MSK;
vector unsigned int w, x, y, z;

z = a->s;
w = lung->s;
x = vec_perm(w, (vector unsigned int)perm, perm);
y = vec_perm(z, sl1_perm, sl1_perm);
y = vec_sll(y, sl1);
y = vec_and(y, sl1_msk);
w = vec_xor(x, b->s);
w = vec_xor(w, y);
x = vec_perm(w, (vector unsigned int)sr1_perm, sr1_perm);
x = vec_srl(x, sr1);
x = vec_and(x, sr1_msk);
y = vec_and(w, msk1);
z = vec_xor(z, y);
r->s = vec_xor(z, x);
lung->s = w;
}
#endif

#elif defined(HAVE_SSE2)
/**
 * This function setup some constant variables for SSE2.
 */
static void setup_const(void) {
static int first = 1;
if (!first) {

}
return;
}
sse2_param_mask = _mm_set_epi32(DSFMT_MSK32_3, DSFMT_MSK32_4,
DSFMT_MSK32_1, DSFMT_MSK32_2);
sse2_int_one = _mm_set_epi32(0, 1, 0, 1);
sse2_double_two = _mm_set_pd(2.0, 2.0);
sse2_double_m_one = _mm_set_pd(-1.0, -1.0);
first = 0;
}

/**
* This function represents the recursion formula.
* @param r output 128-bit
* @param a a 128-bit part of the internal state array
* @param b a 128-bit part of the internal state array
* @param d a 128-bit part of the internal state array (I/O)
*/
inline static void do_recursion(w128_t *r, w128_t *a, w128_t *b, w128_t *u) {
___m128i v, w, x, y, z;

x = a->si;
z = __mm_slli_epi64(x, DSFMT_SL1);
y = _mm_shuffle_epi32(u->si, SSE2_SHUFF);
z = _mm_xor_si128(z, b->si);
y = _mm_xor_si128(y, z);

v = _mm_srli_epi64(y, DSFMT_SR);
w = _mm_and_si128(y, sse2_param_mask);
v = _mm_xor_si128(v, x);
v = _mm_xor_si128(v, w);
r->si = v;
u->si = y;
}
#endif /* standard C */
/**
* This function represents the recursion formula.
* @param r output 128-bit
* @param a a 128-bit part of the internal state array
* @param b a 128-bit part of the internal state array
* @param d a 128-bit part of the internal state array (I/O)
*/
inline static void do_recursion(w128_t *r, w128_t *a, w128_t *b,
w128_t *lung) {
uint64_t t0, t1, L0, L1;

t0 = a->u[0];
t1 = a->u[1];
L0 = lung->u[0];
L1 = lung->u[1];
lung->u[0] = (t0 << DSFMT_SL1) ^ (L1 >> 32) ^ (L1 << 32) ^ b->u[0];
lung->u[1] = (t1 << DSFMT_SL1) ^ (L0 >> 32) ^ (L0 << 32) ^ b->u[1];
r->u[0] = (lung->u[0] >> DSFMT_SR) ^ (lung->u[0] & DSFMT_MSK1) ^ t0;
}
#endif

#if defined(HAVE_SSE2)
/**
 * This function converts the double precision floating point numbers which
 * distribute uniformly in the range [1, 2) to those which distribute uniformly
 * in the range [0, 1).
 * @param w 128bit stracture of double precision floating point numbers (I/O)
 */
inline static void convert_c0o1(w128_t *w) {
    w->sd = _mm_add_pd(w->sd, sse2_double_m_one);
}
/**
 * This function converts the double precision floating point numbers which
 * distribute uniformly in the range [1, 2) to those which distribute uniformly
 * in the range (0, 1).
 * @param w 128bit structure of double precision floating point numbers (I/O)
 */
inline static void convert_o0c1(w128_t *w) {
    w->sd = _mm_sub_pd(sse2_double_two, w->sd);
}
/**
 * This function converts the double precision floating point numbers which
 * distribute uniformly in the range [1, 2) to those which distribute uniformly
 * in the range (0, 1).
 * @param w 128bit structure of double precision floating point numbers (I/O)
 */
inline static void convert_o0o1(w128_t *w) {
    w->si = _mm_or_si128(w->si, sse2_int_one);
    w->sd = _mm_add_pd(w->sd, sse2_double_m_one);
}
#else /* standard C and altivec */
/**
 * This function converts the double precision floating point numbers which
 * distribute uniformly in the range [1, 2) to those which distribute uniformly
 * in the range [0, 1).
 */
#endif

114
* @param w 128bit structure of double precision floating point numbers (I/O)
* /
inline static void convert_c001(w128_t *w) {
    w->d[0] = 1.0;
    w->d[1] = 1.0;
}

/**
 * This function converts the double precision floating point numbers which
 * distribute uniformly in the range [1, 2) to those which distribute uniformly
 * in the range (0, 1).
 * @param w 128bit structure of double precision floating point numbers (I/O)
 */
inline static void convert_o0c1(w128_t *w) {
    w->d[0] = 2.0 - w->d[0];
    w->d[1] = 2.0 - w->d[1];
}

/**
 * This function converts the double precision floating point numbers which
 * distribute uniformly in the range [1, 2) to those which distribute uniformly
 * in the range (0, 1).
 * @param w 128bit structure of double precision floating point numbers (I/O)
 */
inline static void convert_o0o1(w128_t *w) {
    w->u[0] |= 1;
    w->u[1] |= 1;
    w->d[0] = 1.0;
    w->d[1] = 1.0;
}
#endif

/**
 * This function fills the user-specified array with double precision
 * floating point pseudorandom numbers of the IEEE 754 format.
 * @param dsfmt dsfmt state vector.
 * @param array an 128-bit array to be filled by pseudorandom numbers.
 * @param size number of 128-bit pseudorandom numbers to be generated.
 */
inline static void gen_rand_array_c1o2(dsfmt_t *dsfmt, w128_t array[],
int size) {
    int i, j;
    w128_t lung;
    lung = dsfmt->status[DSFMT_N];
    do_recursion(&array[0], &dsfmt->status[0], &dsfmt->status[DSFMT_POS1],

115
** This function fills the user-specified array with double precision
* floating point pseudorandom numbers of the IEEE 754 format.
* @param dsfmt dsfmt state vector.
* @param array an 128-bit array to be filled by pseudorandom numbers.
* @param size number of 128-bit pseudorandom numbers to be generated.
*/
inline static void gen_rand_array_c0o1(dsfmt_t *dsfmt, w128_t array[], int size) {
    int i, j;
    w128_t lung;

    lung = dsfmt->status[DSFMT_N];
    do_recursion(&array[0], &dsfmt->status[0], &dsfmt->status[DSFMT_POS1], &lung);
    for (i = 1; i < DSFMT_N - DSFMT_POS1; i++) {
        do_recursion(&array[i], &dsfmt->status[i], &array[i + DSFMT_POS1], &lung);
    }
    for (; i < DSFMT_N; i++) {
        do_recursion(&array[i], &dsfmt->status[i], &array[i + DSFMT_POS1 - DSFMT_N], &lung);
    }
    for (; i < size - DSFMT_N; i++) {
        do_recursion(&array[i], &array[i - DSFMT_N], &array[i + DSFMT_POS1 - DSFMT_N], &lung);
    }
    for (j = 0; j < 2 * DSFMT_N - size; j++) {
        dsfmt->status[j] = array[j + size - DSFMT_N];
    }
    for (; i < size; i++, j++) {
        do_recursion(&array[i], &array[i - DSFMT_N], &array[i + DSFMT_POS1 - DSFMT_N], &lung);
        dsfmt->status[j] = array[i];
    }
    dsfmt->status[DSFMT_N] = lung;
}
for (; i < size - DSFMT_N; i++) {
    do_recursion(&array[i], &array[i - DSFMT_N],
    &array[i + DSFMT_POS1 - DSFMT_N], &lung);
    convert_c0o1(&array[i - DSFMT_N]);
}
for (j = 0; j < 2 * DSFMT_N - size; j++) {
    dsfmt->status[j] = array[j + size - DSFMT_N];
}
for (; i < size; i++, j++) {
    do_recursion(&array[i], &dsfmt->status[i], &dsfmt->status[i + DSFMT_POS1], &lung);
    dsfmt->status[j] = array[i];
    convert_c0o1(&array[i - DSFMT_N]);
}
for (i = size - DSFMT_N; i < size; i++) {
    convert_c0o1(&array[i]);
} }
dsfmt->status[DSFMT_N] = lung;

/**
 * This function fills the user-specified array with double precision
 * floating point pseudorandom numbers of the IEEE 754 format.
 * @param dsfmt dsfmt state vector.
 * @param array an 128-bit array to be filled by pseudorandom numbers.
 * @param size number of 128-bit pseudorandom numbers to be generated.
 */
inline static void gen_rand_array_o0o1(dsfmt_t *dsfmt, w128_t array[],
    int size) {
    int i, j;
w128_t lung;

    lung = dsfmt->status[DSFMT_N];
do_recursion(&array[0], &dsfmt->status[0], &dsfmt->status[DSFMT_POS1],
    &lung);
for (i = 1; i < DSFMT_N - DSFMT_POS1; i++) {
    do_recursion(&array[i], &dsfmt->status[i],
    &dsfmt->status[i + DSFMT_POS1], &lung);
}
for (; i < DSFMT_N; i++) {
    do_recursion(&array[i], &dsfmt->status[i],
    &array[i + DSFMT_POS1 - DSFMT_N], &lung);
}
for (; i < size - DSFMT_N; i++) {
    do_recursion(&array[i], &array[i - DSFMT_N],
    &array[i + DSFMT_POS1 - DSFMT_N], &lung);
}
```c
for (j = 0; j < 2 * DSFMT_N - size; j++) {
    dsfmt->status[j] = array[j + size - DSFMT_N];
}
for (; i < size; i++, j++) {
    do_recursion(&array[i], &array[i - DSFMT_N],
                 &array[i + DSFMT_POS1 - DSFMT_N], &lung);
    dsfmt->status[j] = array[i];
    convert_o0o1(&array[i - DSFMT_N]);
}
for (i = size - DSFMT_N; i < size; i++) {
    convert_o0o1(&array[i]);
}
}
```

/**
* This function fills the user-specified array with double precision
* floating point pseudorandom numbers of the IEEE 754 format.
* @param dsfmt dsfmt state vector.
* @param array an 128-bit array to be filled by pseudorandom numbers.
* @param size number of 128-bit pseudorandom numbers to be generated.
*/
inline static void gen_rand_array_o0c1(dsfmt_t *dsfmt, w128_t array[], int size) {
    int i, j;
    w128_t lung;
    lung = dsfmt->status[DSFMT_N];
    do_recursion(&array[0], &dsfmt->status[0], &dsfmt->status[DSFMT_POS1],
                 &lung);
    for (i = 1; i < DSFMT_N - DSFMT_POS1; i++) {
        do_recursion(&array[i], &dsfmt->status[i],
                     &dsfmt->status[i + DSFMT_POS1], &lung);
    }
    for (; i < DSFMT_N; i++) {
        do_recursion(&array[i], &dsfmt->status[i],
                     &array[i + DSFMT_POS1 - DSFMT_N], &lung);
    }
    for (; i < size - DSFMT_N; i++) {
        do_recursion(&array[i], &array[i - DSFMT_N],
                     &array[i + DSFMT_POS1 - DSFMT_N], &lung);
        convert_o0c1(&array[i - DSFMT_N]);
    }
```
for (j = 0; j < 2 * DSFMT_N - size; j++) {
    dsfmt->status[j] = array[j + size - DSFMT_N];
}
for (; i < size; i++, j++) {
    do_recursion(&array[i], &array[i - DSFMT_N],
                  &array[i + DSFMT_POS1 - DSFMT_N], &lung);
    dsfmt->status[j] = array[i];
    convert_o0c1(&array[i - DSFMT_N]);
}
for (i = size - DSFMT_N; i < size; i++) {
    convert_o0c1(&array[i]);
    dsfmt->status[DSFMT_N] = lung;
}

/**
 * This function represents a function used in the initialization
 * by init_by_array
 * @param x 32-bit integer
 * @return 32-bit integer
 */
static uint32_t ini_func1(uint32_t x) {
    return (x ^ (x >> 27)) * (uint32_t)1664525UL;
}

/**
 * This function represents a function used in the initialization
 * by init_by_array
 * @param x 32-bit integer
 * @return 32-bit integer
 */
static uint32_t ini_func2(uint32_t x) {
    return (x ^ (x >> 27)) * (uint32_t)1566083941UL;
}

/**
 * This function initializes the internal state array to fit the IEEE
 * 754 format.
 * @param dsfmt dsfmt state vector.
 */
static void initial_mask(dsfmt_t *dsfmt) {
    int i;
    uint64_t *psfmt = &dsfmt->status[0].u[0];
    for (i = 0; i < DSFMT_N * 2; i++) {
        
119
psfmt[i] = (psfmt[i] & DSFMT_LOW_MASK) | DSFMT_HIGH_CONST;
}

/**
 * This function certificate the period of 2^{SFMT_MEXP} -1.
 * @param dsfmt dsfmt state vector.
 */
static void period_certification(dsfmt_t *dsfmt) {
    uint64_t pcv[2] = {DSFMT_PCV1, DSFMT_PCV2};
    uint64_t tmp[2];
    uint64_t inner;

    tmp[0] = (dsfmt->status[DSFMT_N].u[0] ^ DSFMT_FIX1);
    tmp[1] = (dsfmt->status[DSFMT_N].u[1] ^ DSFMT_FIX2);

    inner = tmp[0] & pcv[0];
    inner ^= tmp[1] & pcv[1];
    for (int i = 32; i > 0; i >>= 1) {
        inner ^= inner >> i;
    }
    inner &= 1;
    /* check OK */
    if (inner == 1) {
        return;
    }
    /* check NG, and modification */
    #if (DSFMT_PCV2 & 1) == 1
    dsfmt->status[DSFMT_N].u[1] ^= 1;
    #else
    uint64_t work;
    for (int i = 1; i >= 0; i--) {
        work = 1;
        for (int j = 0; j < 64; j++) {
            if (((work & pcv[i]) != 0) {
                dsfmt->status[DSFMT_N].u[i] ^= work;
                return;
            }
            work = work << 1;
        }
    }
    #endif
    return;
}

/****************************
PUBLIC FUNCTIONS
-------------------*/
/**
* This function returns the identification string. The string shows
* the Mersenne exponent, and all parameters of this generator.
* @return id string.
*/
const char *dsfmt_get_idstring(void) {
    return DSFMT_IDSTR;
}

/**
* This function returns the minimum size of array used for
* fill_array functions.
* @return minimum size of array used for fill_array functions.
*/
int dsfmt_get_min_array_size(void) {
    return DSFMT_N64;
}

/**
* This function fills the internal state array with double precision
* floating point pseudorandom numbers of the IEEE 754 format.
* @param dsfmt dsfmt state vector.
*/
void dsfmt_gen_rand_all(dsfmt_t *dsfmt) {
    int i;
    w128_t lung;
    lung = dsfmt->status[DSFMT_N];
    do_recursion(&dsfmt->status[0], &dsfmt->status[0],
                 &dsfmt->status[DSFMT_POS1], &lung);
    for (i = 1; i < DSFMT_N - DSFMT_POS1; i++) {
        do_recursion(&dsfmt->status[i], &dsfmt->status[i],
                     &dsfmt->status[i + DSFMT_POS1], &lung);
    }
    for (; i < DSFMT_N; i++) {
        do_recursion(&dsfmt->status[i], &dsfmt->status[i],
                     &dsfmt->status[i + DSFMT_POS1 - DSFMT_N], &lung);
    }
    dsfmt->status[DSFMT_N] = lung;
}

/**
* This function generates double precision floating point
* pseudorandom numbers which distribute in the range [1, 2) to the
* specified array[] by one call. The number of pseudorandom numbers
* is specified by the argument `size`, which must be at least \((\text{SFMT\_MEXP} /
2) * 2\) and a multiple of two. The function
* `get_min_array_size()` returns this minimum size. The generation by
* this function is much faster than the following `fill_array_xxx` functions.

* For initialization, `init_gen_rand()` or `init_by_array()` must be called
* before the first call of this function. This function cannot be
* used after calling `genrand_xxx` functions, without initialization.

* @param dsfmt dsfmt state vector.
* @param array an array where pseudorandom numbers are filled
* by this function. The pointer to the array must be "aligned"
* (namely, must be a multiple of 16) in the SIMD version, since it
* refers to the address of a 128-bit integer. In the standard C
* version, the pointer is arbitrary.

* @param size the number of 64-bit pseudorandom integers to be
* generated. `size` must be a multiple of 2, and greater than or equal
* to \((\text{SFMT\_MEXP} / 128) * 2\).

* @note \`memalign` or \`posix_memalign` is available to get aligned
* memory. Mac OSX doesn't have these functions, but \`malloc` of OSX
* returns the pointer to the aligned memory block.

*/
void dsfmt_fill_array_close1_open2(dsfmt_t *dsfmt, double array[], int size) {
    assert(size % 2 == 0);
    assert(size >= DSFMT_N64);
    gen_rand_array_c1o2(dsfmt, (w128_t *)array, size / 2);
}

/**
* This function generates double precision floating point
* pseudorandom numbers which distribute in the range \((0, 1]\) to the
* specified array[] by one call. This function is the same as
* `fill_array_close1_open2()` except the distribution range.
*
* @param dsfmt dsfmt state vector.
* @param array an array where pseudorandom numbers are filled
* by this function.
* @param size the number of pseudorandom numbers to be generated.
* see also \`fill_array_close1_open2()`
*/
void dsfmt_fill_array_open_close(dsfmt_t *dsfmt, double array[], int size) {
    assert(size % 2 == 0);
    assert(size >= DSFMT_N64);
gen_rand_array_o0c1(dsfmt, (w128_t *)array, size / 2);
}

/**
 * This function generates double precision floating point 
 * pseudorandom numbers which distribute in the range [0, 1) to the 
 * specified array[] by one call. This function is the same as 
 * fill_array_close1_open2() except the distribution range. 
 * 
 * @param array an array where pseudorandom numbers are filled 
 * by this function. 
 * @param dsfmt dsfmt state vector. 
 * @param size the number of pseudorandom numbers to be generated. 
 * see also \sa fill_array_close1_open2() 
 */
void dsfmt_fill_array_close_open(dsfmt_t *dsfmt, double array[], int size) {
    assert(size % 2 == 0);
    assert(size >= DSFMT_N64);
    gen_rand_array_c0o1(dsfmt, (w128_t *)array, size / 2);
}

/**
 * This function generates double precision floating point 
 * pseudorandom numbers which distribute in the range (0, 1) to the 
 * specified array[] by one call. This function is the same as 
 * fill_array_close1_open2() except the distribution range. 
 * 
 * @param dsfmt dsfmt state vector. 
 * @param array an array where pseudorandom numbers are filled 
 * by this function. 
 * @param size the number of pseudorandom numbers to be generated. 
 * see also \sa fill_array_close1_open2() 
 */
void dsfmt_fill_array_open_open(dsfmt_t *dsfmt, double array[], int size) {
    assert(size % 2 == 0);
    assert(size >= DSFMT_N64);
    gen_rand_array_o001(dsfmt, (w128_t *)array, size / 2);
}

#if defined(__INTEL_COMPILER)
#  pragma warning(disable:981)
#endif

/* This function initializes the internal state array with a 32-bit 
 * integer seed. 
 * @param dsfmt dsfmt state vector. 
 */
/* @param seed a 32-bit integer used as the seed.
 * @param mexp caller's mersenne exponent */

void dsfmt_chk_init_gen_rand(dsfmt_t *dsfmt, uint32_t seed, int mexp) {
    int i;
    uint32_t *psfmt;

    /* make sure caller program is compiled with the same MEXP */
    if (mexp != dsfmt_mexp) {
        fprintf(stderr, "DSFMT_MEXP doesn't match with dSFMT.c\n");
        exit(1);
    }
    psfmt = &dsfmt->status[0].u32[0];
    psfmt[idxof(0)] = seed;
    for (i = 1; i < (DSFMT_N + 1) * 4; i++) {
        psfmt[idxof(i)] = 1812433253UL * (psfmt[idxof(i - 1)] ^ (psfmt[idxof(i - 1)] >> 30)) + i;
    }
    initial_mask(dsfmt);
    period_certification(dsfmt);
    dsfmt->idx = DSFMT_N64;
    #if defined(HAVE_SSE2)
    setup_const();
    #endif
}

/**
 * This function initializes the internal state array,
 * with an array of 32-bit integers used as the seeds
 * @param dsfmt dsfmt state vector.
 * @param init_key the array of 32-bit integers, used as a seed.
 * @param key_length the length of init_key.
 * @param mexp caller's mersenne exponent */

void dsfmt_chk_init_by_array(dsfmt_t *dsfmt, uint32_t init_key[],
                             int key_length, int mexp) {
    int i, j, count;
    int32_t t r;
    uint32_t *psfmt32;
    int lag;
    int mid;
    int size = (DSFMT_N + 1) * 4;   /* pulmonary */

    /* make sure caller program is compiled with the same MEXP */
    if (mexp != dsfmt_mexp) {
        fprintf(stderr, "DSFMT_MEXP doesn't match with dSFMT.c\n");
    }
exit(1);
}
if (size >= 623) {
lag = 11;
} else if (size >= 68) {
lag = 7;
} else if (size >= 39) {
lag = 5;
} else {
lag = 3;
}
mid = (size - lag) / 2;

psfmt32 = &dsfmt->status[0].u32[0];
memset(dsfmt->status, 0x8b, sizeof(dsfmt->status));
if (key_length + 1 > size) {
count = key_length + 1;
} else {
count = size;
}
r = ini_func1(psfmt32[idxof(0)] ^ psfmt32[idxof(mid % size)]
^ psfmt32[idxof((size - 1) % size)));
psfmt32[idxof(mid % size)] += r;
r += key_length;
psfmt32[idxof((mid + lag) % size)] += r;
psfmt32[idxof(0)] = r;
i = 1;
count--;
for (i = 1, j = 0; (j < count) && (j < key_length); j++) {
  r = ini_func1(psfmt32[idxof(i)]
    ^ psfmt32[idxof((i + mid) % size)]
    ^ psfmt32[idxof((i + size - 1) % size)));
  psfmt32[idxof((i + mid) % size)] += r;
  r += init_key[j] + i;
  psfmt32[idxof((i + mid + lag) % size)] += r;
  psfmt32[idxof(i)] = r;
  i = (i + 1) % size;
}
for (; j < count; j++) {
  r = ini_func1(psfmt32[idxof(i)]
    ^ psfmt32[idxof((i + mid) % size)]
    ^ psfmt32[idxof((i + size - 1) % size)));
  psfmt32[idxof((i + mid) % size)] += r;
  r += i;
  psfmt32[idxof((i + mid + lag) % size)] += r;
  psfmt32[idxof(i)] = r;
\begin{verbatim}
    i = (i + 1) % size;
}
for (j = 0; j < size; j++) {
    r = ini_func2(psfmt32[idxof(i)]
        + psfmt32[idxof((i + mid) % size)]
        + psfmt32[idxof((i + size - 1) % size)]);
    psfmt32[idxof((i + mid) % size)] ^= r;
    r ^= i;
    psfmt32[idxof((i + mid + lag) % size)] ^= r;
    psfmt32[idxof(i)] = r;
    i = (i + 1) % size;
}
initial_mask(dsfmt);
period_certification(dsfmt);
dsfmt->idx = DSFMT_N64;
#endif
#endif

float Min(float a, float b)
{
    if (a-b<0.0) return a;
    else return b;
}

/* MAIN FUNCTION */

int main()
{
    register short i, j;       // counters commonly used in 'for' loops
    register short atomico;    // chosen atom to be moved. defined as register.
    register const short ciminimum_distance_square = 3215; // criteria for overlap of atoms
    struct tm *local;
    time_t t;
    double dV[g_csatomnoplus][g_csatomnoplus];
    double dV_temp[g_csatomnoplus][g_csatomnoplus];
    double dratio8[g_csatomnoplus][g_csatomnoplus];
    double dratio8_temp[g_csatomnoplus][g_csatomnoplus];
    double drho[g_csatomnoplus];
    double drho_temp[g_csatomnoplus];
    const double hundredthLC = LATTICECONSTANT/100.0;
\end{verbatim}
const double cdlowelimit = -525.0;  // low energy limit
cost double cdhighelimit = 0.1*((floor)(8.0*cdlowelimit));  // high energy limit
const double cdspacing = 0.1;      // span of a energy bin
const int cinofbin = floor((float)((cdhighelimit-cdlowelimit)/cdspacing));
unsigned long long int llihistogram[cinofbin];  // histogram height

double GAMMA = 0.01;
double dlnge[cinofbin];            // natural logarithm of density of states
double dlnge_agu[cinofbin];
double dlnge_guinc[cinofbin];
int isizeofbox;                    // half of the simulation box size

/*. Sutton-Chen parameters lattice constant a = 4.08(LATTICACONSTANT), exponent for attractive m=8, for repulsive n=10 */
const double dc = 53.581;          // Sutton-Chen parameter for Au
const double depsilon = 0.0078052; // Sutton-Chen parameter for Au

double denergyi;                  // energy of the system before the change in configuration
double dalpha;                    // variable used to store the value of lowest energy configuration
short sbini;                      // energy bin before the change in configuration
int m;
int ilocalcounter;                // counter for accepted local moves
double ddeltaE;                   // energy difference before and after a localmove
int im, in, ip, idm, idn, idp;     // distances along an axis
const double tenthousand = 10000.0;
double dratio2, dratio3;

double denergyf;                  // energy of the system after the change in the configuration
short sbinf;                      // energy bin after the change in configuration
int isumx, isumy, isumz;          // sum of coordinates in order to get the center of mass
struct latticestruct icm;         // the center of mass
int imaxrfromcm;                  // maximum distance square
int irfromcm;                     // distance square from the center of mass
float fmaxrfromcm;                // maximum distance = sqrt(imaxrfromcm)
float ftheta1, ftheta2;           // auxiliary variables to set a new position with surface move
int iguc = 0;
_Bool bdogu = 0;
_Bool brepeat = 0;                // 0 or 1 to determine repetition after histogram flatness check
double dlngediff;
short slowbin;
unsigned long long int llihistsum; // sum of histogram heights (=number of random walks performed)
short snofhistbin;
float fhistave;                   // average height of histogram (=dhistsum/cinofbin)

// Opening files
filecrystal=fopen("/home/kingrice/bin/SC_Q_G16OMP_N150-525.0-0.8E0_2box_configh.dat","w");  // This file contains the lowest energy configuration found
filedos=fopen("/home/kingrice/bin/SC_Q_G16OMP_N150-525.0-0.8E0_2box_dosh.dat","w"); // This file contains the density of states

// Printing time
int t = time(NULL);
time_t local = localtime(&t);
fprintf(filedos,"Start time : %s\n", asctime(local));

// Initialization of arrays. CHECKED OK
Initialize_arrays

// Reading initial coordinate from a pdb file and assign each particle to a lattice site.
CHECKED OK
Readinitialcoordinates();

// Calculation of the simulation box size. CHECKED OK
printf("SC%d ",ATOMNO);
isizeofbox=Setboxsize();

// Evaluation of the initial distances. CHECKED OK
Distances();

// Evaluation of the initial energy. CHECKED OK
denergyi=depsilon*EnergyInitial (dV,dratio8,drho,dc);

deenergyi=denergyi; // initializes the variable that tracks the minimum energy found
sbini=floor((denergyi-cdlowelimit)/cdspacing);
printf("Initial Energy is %lf and corresponding bin is [\%d]\n",denergyi,sbini);

if(sbini-cinofbin>0||sbini<0){
    printf("Error in bin number calculation!!! Aborted!!!\n");
    exit(1);
}

//printf("Exiting at line 1075\n");
exi	(0);
dsfmt_init_gen_rand(&dsfmt,ATOMNO*isizeofbox+10);

do
 {
  do
   {
    do
      {
        //PARALLELIZED BLOCK BEGINS
        #pragma omp parallel for schedule(static) default(shared)
        private(m,ilocalcounter,atomico,g_ilatticeatom_aux,ddeltaE,drho_temp,im,in,ip,i,j,idm,idn,idp,g_idist_atom_aux,dratio2,dratio3,dratio8_temp,dV_temp,denergyf,sbini,isumx,isumz,icm,imaxrfromcm,irfromcm,fmaxrfromcm,ftheta1,ftheta2) \ 
        firstprivate(g_ilatticeatom,g_idist_atom,dratio8,drho,dV,sbini,denergyi) \ 
        lastprivate(g_ilatticeatom,g_idist_atom,dratio8,drho,dV,sbini,denergyi)
        for(m=1;m<=200000;m++)
        {
          ilocalcounter=0;
          do /* Local move starts here */
          {
            // Selection of an atom to be moved
            atomico=1+floor(dsfmt_genrand_close_open(&dsfmt)*ATOMNO);
            // Displacement of the selected atom
            // Localmove()
            g_ilatticeatom_aux[atomico].x=g_ilatticeatom[atomico].x - 3 + floor(dsfmt_genrand_close_open(&dsfmt)*7.0);
            g_ilatticeatom_aux[atomico].y=g_ilatticeatom[atomico].y - 3 + floor(dsfmt_genrand_close_open(&dsfmt)*7.0);
            g_ilatticeatom_aux[atomico].z=g_ilatticeatom[atomico].z - 3 + floor(dsfmt_genrand_close_open(&dsfmt)*7.0);

            // Check the selected atom is still inside the box after the displacement.
            if (((1+((isizeofbox-abs(g_ilatticeatom_aux[atomico].x))>>31))*(1+((isizeofbox-abs(g_ilatticeatom_aux[atomico].y))>>31))*(1+((isizeofbox-abs(g_ilatticeatom_aux[atomico].z))>>31)))) // if inside the box
            {
              // EnergyDifference()
              ddeltaE=0.0;
              drho_temp[atomico]=0.0;
              im=-g_ilatticeatom_aux[atomico].x;
              in=-g_ilatticeatom_aux[atomico].y;
            }
ip=-g_ilatticeatom_aux[atomico].z;
i=atomico-1;
j=1;
for(i; i--; ) // reverse loop for speedup
{
    j=i+1; // each atom in the Verlet list
    idm=(im+g_ilatticeatom[j].x)*(im+g_ilatticeatom[j].x);
    idn=(in+g_ilatticeatom[j].y)*(in+g_ilatticeatom[j].y);
    idp=(ip+g_ilatticeatom[j].z)*(ip+g_ilatticeatom[j].z);
    g_idist_atom_aux[atomico][j]=idm+idn+idp;
    if(g_idist_atom_aux[atomico][j]-ciminimum_distance_square<0) // Overlap criteria
    {
        ddeltaE=123456789.0;
        j=0;
        break;
    }
    dratio2=thousandth/g_idist_atom_aux[atomico][j];
    dratio3=dratio2*sqrt(dratio2); 
    dratio8_temp[atomico][j]=dratio3*dratio3*dratio2; 
    drho_temp[atomico]+=dratio8_temp[atomico][j]; // for new attractive energy from "atomico"
    drho_temp[j]=drho[j]-
    dratio8[atomico][j]+dratio8_temp[atomico][j]; 
    ddeltaE+=dc*(sqrt(drho[j])-sqrt(drho_temp[j])); // subtracting old attractive energy & adding new attractive energy
    dV_temp[atomico][j]=dratio3*dratio8_temp[atomico][j];
    ddeltaE+=dV_temp[atomico][j]-dV[atomico][j]; // adding new repulsive energy & subtracting old repulsive energy
}
if(j!=0)
{
    i=ATOMNO-atomico;
    for(i--; ) // reverse loop for speedup
    {
        j=i+atomico+1; // each atom in the Verlet list
        idm=(im+g_ilatticeatom[j].x)*(im+g_ilatticeatom[j].x);
        idn=(in+g_ilatticeatom[j].y)*(in+g_ilatticeatom[j].y);
        idp=(ip+g_ilatticeatom[j].z)*(ip+g_ilatticeatom[j].z);
        g_idist_atom_aux[atomico][j]=idm+idn+idp;
        if(g_idist_atom_aux[atomico][j]-ciminimum_distance_square<0) // Overlap criteria
        {
            ddeltaE=123456789.0;
            break;
        }
    }
}
dratio2=tenthousand/g_idist_atom_aux[atomico][j];

dratio3=dratio2*sqrt(dratio2);

dratio8_temp[atomico][j]=dratio3*dratio3*dratio2;

drho_temp[atomico]+=dratio8_temp[atomico][j]; // for new attractive energy from "atomico"

drho_temp[j]=drho[j]-

dratio8[atomico][j]+dratio8_temp[atomico][j];

ddeltaE+=dc*(sqrt(drho[j])-sqrt(drho_temp[j])); // subtracting old attractive energy & adding new attractive energy

dV_temp[atomico][j]=dratio3*dratio8_temp[atomico][j];

ddeltaE+=(dV_temp[atomico][j]-dV[atomico][j]); // adding new repulsive energy & subtracting old repulsive energy

if(0.0==ddeltaE-123456789.0) // if overlapped
{
  llihistogram[sbini]+=1.0;
  dlnge[sbini]+=GAMMA;
  #pragma omp flush(dlnge)
}

else // if not overlapped
{
  ddeltaE+=dc*(sqrt(drho[atomico])-sqrt(drho_temp[atomico]));
  // subtracting & adding attractive energy from "atomico"
  ddeltaE+=depsilon;
  denergyf = denergyi + ddeltaE;

  if(0.0<=denergyf-cdhighelimit) // high energy limit
  {
    llihistogram[sbini]+=1.0;
    dlnge[sbini]+=GAMMA;
    #pragma omp flush(dlnge)
  }

  else
  {
    if(0.0>denergyf-cdlowelimit) // low energy limit
    {
      llihistogram[sbini]+=1.0;
      dlnge[sbini]+=GAMMA;
      #pragma omp flush(dlnge)
    }

    else
    {
      if(0.0>denergyf-dalpha) // low energy limit
      {
        dalpha=denergyf;
      }

      #pragma omp flush(dalpha)
    }
  }

}
fprintf(filecrystal,"A\nB\nC\nEnergy outside the window by a local move is %lf\n",denergyf);
fprintf(filecrystal,"Coordinate of atoms\n");
for(i=1;i<atomico;i++)
    fprintf(filecrystal,"ATOM %3d Au MOL 0
0\n",i,hundredthLC*g_ilatticeatom[i].x,hundredthLC*g_ilatticeatom[i].y,hundredthLC*g_ilatticeatom[i].z);
fprintf(filecrystal,"ATOM %3d Au MOL 0
0\n",i,hundredthLC*g_ilatticeatom_aux[atomico].x,hundredthLC*g_ilatticeatom_aux[atomico].y,hundredthLC*g_ilatticeatom_aux[atomico].z);
for(i=atomico+1;i<=ATOMNO;i++)
    fprintf(filecrystal,"ATOM %3d Au MOL 0
0\n",i,hundredthLC*g_ilatticeatom[i].x,hundredthLC*g_ilatticeatom[i].y,hundredthLC*g_ilatticeatom[i].z);
fprintf(filecrystal,"TER\n");
fflush(filecrystal);
}
else // in the energy window
{
   sbinf=floor((denergyf-cdlowelimit)/cdspacing);
    if(0.0>dsfmt_genrand_close_open(&dsfmt)-Min(1.0,exp(dlnge[sbinf]-dlnge[sbinf]))) // WL accepted
    {
        llihistogram[sbinf]+=1.0;
        dlnge[sbinf]+=GAMMA;
        #pragma omp flush(dlnge)
        ilocalcounter++;
        // Energy is updated
denergyi=denergyf;
sbin=sbinf;
        // Coordinates are updated
g_ilatticeatom[atomico].x=g_ilatticeatom_aux[atomico].x;
g_ilatticeatom[atomico].y=g_ilatticeatom_aux[atomico].y;
g_ilatticeatom[atomico].z=g_ilatticeatom_aux[atomico].z;
        // Distances and energy matrices are updated
        i=ATOMNO;
        for(i--;)
        {
            j=i+1;
g_idist_atom[atomico][j]=g_idist_atom_aux[atomico][j];

g_idist_atom[j][atomico]=g_idist_atom[atomico][j];

dV[atomico][j]=dV_temp[atomico][j];
dV[j][atomico]=dV[atomico][j];

dratio8[atomico][j]=dratio8_temp[atomico][j];
dratio8[j][atomico]=dratio8[atomico][j];

drho[j]=drho_temp[j];

} // Storing energy and coordinates of lowest energy state found

if(0.0>denergyf-dalpha)
{
  dalpha=denergyf;
  #pragma omp flush(dalpha)
  fprintf(filecrystal,"AnBnCnnThe lowest energy found by local moves is %lf[%d]n",denergyf,sbinf);
  fprintf(filecrystal,"Coordinate of atoms\n");
  for(i=1;i<=ATOMNO;i++)
    fprintf(filecrystal,"ATOM %3d   Au MOL 0 1%8.3lf%8.3lf%8.3lf 1.00 0.00          Au 0\n",i,hundredthLC*g_ilatticeatom[i].x,hundredthLC*g_ilatticeatom[i].y,hundredthLC*g_ilatticeatom[i].z);
  fprintf(filecrystal,"TER\n");
  fflush(filecrystal);
}
else // WL rejected
{
  llihistogram[sbini]+=1.0;
  dlnge[sbini]+=GAMMA;
  #pragma omp flush(dlnge)
}

} // above Elow, thus in the E window
} // below Ehigh
} // no overlap
} // inside the box
else // outside the box
{
  llihistogram[sbini]+=1.0;
  dlnge[sbini]+=GAMMA;
}
#pragma omp flush(dlnge)
}
}
while (ilocalcounter-10000<0);

/* Surface move starts here */
// Calculation of center of mass and initialization of matrix
g_idist_atom_temp
isumx=0; isumy=0; isumz=0;
i=ATOMNO;
for(;i--;
{
    j=i+1;
    isumx+=g_ilatticeatom[j].x;
    isumy+=g_ilatticeatom[j].y;
    isumz+=g_ilatticeatom[j].z;
}
//printf("sumx=%d sumy=%d sumz=%d\n",isumx,isumy,isumz);
icm.x=floor(isumx/ATOMNO);
icm.y=floor(isumy/ATOMNO);
icm.z=floor(isumz/ATOMNO);
//printf("x=%d y=%d z=%d\n",icm.x,icm.y,icm.z);
// Selection of the fairest atom from the CM
imaxrfromcm=1;
i=ATOMNO;
for(;i--;
{
    j=i+1;
    im=(g_ilatticeatom[j].x-icm.x)*(g_ilatticeatom[j].x-icm.x);
    in=(g_ilatticeatom[j].y-icm.y)*(g_ilatticeatom[j].y-icm.y);
    ip=(g_ilatticeatom[j].z-icm.z)*(g_ilatticeatom[j].z-icm.z);
    irfromcm=im+in+ip;
    if(imaxrfromcm-irfromcm<0)
        {
            imaxrfromcm=irfromcm;
            atomico=j;
        }
}
// Random displacement along surface
fmaxrfromcm=sqrt(imaxrfromcm);
fttheta1=PI*dsfmt_genrand_close_open(&dsfmt);
fttheta2=2.0*PI*dsfmt_genrand_close_open(&dsfmt);

g_ilatticeatom_aux[atomico].x=floor(fmaxrfromcm*sin(fttheta1)*cos(fttheta2))+icm.x;
g_ilatticeatom_aux[atomico].y=floor(fmaxrfromcm*sin(fttheta1)*sin(fttheta2))+icm.y;
g_ilatticeatom_aux[atomico].z = floor(fmaxfromcm * cos(ftheta1)) + icm.z;

// Check if still inside the box
if (((1+((isizeofbox- 
abs(g_ilatticeatom_aux[atomico].x))>>31))*(1+((isizeofbox- 
abs(g_ilatticeatom_aux[atomico].y))>>31))*(1+((isizeofbox- 
abs(g_ilatticeatom_aux[atomico].z))>>31)))
{
    // EnergyDifference()
    ddeltaE = 0.0;
    drho_temp[atomico] = 0.0;
    im = -g_ilatticeatom_aux[atomico].x;
    in = -g_ilatticeatom_aux[atomico].y;
    ip = -g_ilatticeatom_aux[atomico].z;
    i = atomico - 1;
    j = 1;
    for(; i--; ) // reverse loop for speedup
    {
        j = i + 1; // each atom in the Verlet list
        idm = (im + g_ilatticeatom[j].x) * (im + g_ilatticeatom[j].x);
        idn = (in + g_ilatticeatom[j].y) * (in + g_ilatticeatom[j].y);
        idp = (ip + g_ilatticeatom[j].z) * (ip + g_ilatticeatom[j].z);
        g_idist_atom_aux[atomico][j] = idm + idn + idp;
        if (g_idist_atom_aux[atomico][j] - minimum_distance_square < 0)
            // Overlap criteria
            { 
                ddeltaE = 123456789.0;
                j = 0;
                break;
            }
        dratio2 = ten thousand / g_idist_atom_aux[atomico][j];
        dratio3 = dratio2 * sqrt(dratio2);
        dratio8_temp[atomico][j] = dratio3 * dratio3 * dratio2;
        drho_temp[atomico] += dratio8_temp[atomico][j]; // for new attractive energy from "atomico"
        drho_temp[j] = drho[j] - dratio8[atomico][j] + dratio8_temp[atomico][j];
        ddeltaE += dc * (sqrt(drho[j]) - sqrt(drho_temp[j])); // subtracting old attractive energy & adding new attractive energy
        dV_temp[atomico][j] = dratio3 * dratio8_temp[atomico][j];
        ddeltaE += (dV_temp[atomico][j] - dV[atomico][j]); // adding new repulsive energy & subtracting old repulsive energy
    }
    if (j != 0)
    {
        i = ATOMNO - atomico;
    }
for(;i--;)// reverse loop for speedup
{
    j=i+atomico+1;// each atom in the Verlet list
    idm=(im+g_ilatticeatom[j].x)*(im+g_ilatticeatom[j].x);
    idn=(in+g_ilatticeatom[j].y)*(in+g_ilatticeatom[j].y);
    idp=(ip+g_ilatticeatom[j].z)*(ip+g_ilatticeatom[j].z);
    g_idist_atom_aux[atomico][j]=idm+idn+idp;
    if(g_idist_atom_aux[atomico][j] - ciminimum_distance_square<0) // Overlap criteria
    {
        ddeltaE=123456789.0;
        break;
    }
    dratio2=tenthousand/g_idist_atom_aux[atomico][j];
    dratio3=dratio2*sqrt(dratio2);
    dratio8_temp[atomico][j]=dratio3*dratio3*dratio2;
    drho_temp[atomico]+=dratio8_temp[atomico][j]; // for new attractive energy from "atomico"
    drho_temp[j]=drho[j]-
    dratio8[atomico][j]+dratio8_temp[atomico][j];
    ddeltaE+=dc*(sqrt(drho[j])-sqrt(drho_temp[j])); // subtracting old attractive energy & adding new attractive energy
    dV_temp[atomico][j]=dratio3*dratio8_temp[atomico][j];
    ddeltaE+=(dV_temp[atomico][j]-dV[atomico][j]); // adding new repulsive energy & subtracting old repulsive energy
}

if(0.0==ddeltaE-123456789.0)// if overlapped
{
    llihistogram[sbini]+=1.0;
    dlnge[sbini]+=GAMMA;
    #pragma omp flush(dlnge)
}
else // if not overlapped
{
    ddeltaE+=dc*(sqrt(drho[atomico])-sqrt(drho_temp[atomico])); // subtracting & adding attractive energy from "atomico"
    ddeltaE*=depsilon;
    denergyf = denergyi + ddeltaE;

    if(0.0<=denergyf-cdhighelimit)// high energy limit
    {
        llihistogram[sbini]+=1.0;
        dlnge[sbini]+=GAMMA;
        #pragma omp flush(dlnge)
    }
} 
else 
{
    if(0.0>denergyf-cdlowlimit) // low energy limit
    {
        llii_histogram[sbini]+=1.0;
        dlnge[sbini]+=GAMMA;
    }
} 
#pragma omp flush(dlnge)
if(0.0>denergyf-dalpha) 
{
    dalpha=denergyf;
}
#pragma omp flush(dalpha)
    fprintf(filecrystal,"\nEnergy outside window by a surface move is %lf\n",denergyf);
    fprintf(filecrystal,"Coordinate of atoms\n");
    for(i=1;i<atomico;i++)
        fprintf(filecrystal,"ATOM %3d   Au MOL 0
0\n",i,hundredthLC*g_ilatticeatom[i].x,hundredthLC*g_ilatticeatom[i].y,hundredthLC*g_ilatticeatom[i].z);
    fprintf(filecrystal,"ATOM %3d   Au MOL 0
0\n",i,hundredthLC*g_ilatticeatom_aux[atomico].x,hundredthLC*g_ilatticeatom_aux[atomico].y,hundredthLC*g_ilatticeatom_aux[atomico].z);
    for(i=atomico+1;i<=ATOMNO;i++)
        fprintf(filecrystal,"ATOM %3d   Au MOL 0
0\n",i,hundredthLC*g_ilatticeatom[i].x,hundredthLC*g_ilatticeatom[i].y,hundredthLC*g_ilatticeatom[i].z);
    fprintf(filecrystal,"TER\n");
    fflush(filecrystal);
} 
else // in the energy window 
{
    sbinf=floor((denergyf-cdlowlimit)/cdspacing);
    if(0.0>dsfmt_genrand_close_open(&dsfmt)-Min(1.0,exp(dlnge[sbini]-dlnge[sbinf]))) // WL accepted
    {
        llii_histogram[sbini]+=1.0;
        dlnge[sbini]+=GAMMA;
    }
    #pragma omp flush(dlnge)
    // Energy is updated
denergi=senergyf;
sbini=sbinf;
    // Coordinates are updated
}
g_ilatticeatom[atomico].x=g_ilatticeatom_aux[atomico].x;
g_ilatticeatom[atomico].y=g_ilatticeatom_aux[atomico].y;
g_ilatticeatom[atomico].z=g_ilatticeatom_aux[atomico].z;
    // Distances and energy matrices are updated
    i=ATOMNO;
    for(;i--;
    {
        j=i+1;
        g_idist_atom[atomico][j]=g_idist_atom_aux[atomico][j];
        g_idist_atom[j][atomico]=g_idist_atom[atomico][j];
        dV[atomico][j]=dV_temp[atomico][j];
        dV[j][atomico]=dV[atomico][j];
        dratio8[atomico][j]=dratio8_temp[atomico][j];
        dratio8[j][atomico]=dratio8[atomico][j];
        drho[j]=drho_temp[j];
    }
    // Save coordinates if lowest energy were found
    if(0.0>denergyf-dalpha)
    {
        dalpha=denergyf;
        #pragma omp flush(dalpha)
        fprintf(filecrystal,"A\nB\nC\nThe lowest energy
found by a surface move is %lf[%d]\n",denergyf,sbinf);
        fprintf(filecrystal,"Coordinate of atoms\n");
        for(i=1;i<=ATOMNO;i++)
            fprintf(filecrystal,"ATOM  %3d  Au MOL 0
0\n",i,hundredthLC*g_ilatticeatom[i].x,hundredthLC*g_ilatticeatom[i].y,hundredthLC*g
_ilatticeatom[i].z);
            fprintf(filecrystal,"TER\n");
        fflush(filecrystal);
    }
else // WL rejected
{
    llihistogram[sbini]+=1.0;
    dlnge[sbini]+=GAMMA;
    #pragma omp flush(dlnge)
else // outside the box
{
  llihistogram[sbini]+=1.0;
  dlnge[sbini]+=GAMMA;
  #pragma omp flush(dlnge)
}
} // for(;m;)

// Check for uniform growth of DOS, which determines the need of another
global update
if(0==iguc) // criterion for the first global update
{
  bdogu=0;
  i=cinofbin;
  for(;i--;
   
  {  
    if(dlnge[i]-OMEGA>0.0)
   
    {  
      bdogu=1;
      break;
    }
   
  }
}
else // criterion for last global updates
{
  bdogu=1;
  llihistsum=0;
  for(i=cinofbin-1;i>=slowbin;i--)
  llihistsum+=llihistogram[i];
  snofhistbin=cinofbin-slowbin;
  fhistave=llihistsum/snofhistbin;
  for(i=cinofbin-1;i>=slowbin;i--)
  {
    if (fabs((fhistave-llihistogram[i])-0.20*fhistave)>0.0)
      
    {  
      bdogu=0;
      break;
    }
  }
}
while(0==bdogu);

// Initialize histogram. Histogram accumulation begins after each global update has been made
i=cinofbin;
for(i--;)
    llihistogram[i]=0;
// If global updates are done for every bin, start refining with smaller value of GAMMA
if(1==brepeat) break;

iguc++;
fprintf(filedos,"DOS before global update %d
\n",iguc);
i=cinofbin;
for(i--;
    fprintf(filedos,"Density of state[%d] = %.3lf
\n",i,dlnge[i]);
    fprintf(filedos,"
\n");
    fflush(filedos);

/* Global update */
i=cinofbin;
for(i--;
    dlngediff=dlnge[i]-OMEGA;
    if(dlngediff>=0.0)
    {
        dlnge_guinc[i]=KAPPA*exp(-LAMBDA/dlngediff);
        dlnge[i]+=dlnge_guinc[i];
        dlnge_agu[i]=dlnge[i];
        slowbin=i;
    }
}

// Printing DOS after global update
printf("Global update %d is performed from bin [%d]\n",iguc,slowbin);
fprintf(filedos,"DOS after global update %d from bin [%d]\n",iguc,slowbin);
i=cinofbin;
for(i--;
    fprintf(filedos,"Density of state[%d] = %.3lf  Increment by GU = %.3lf\n",i,dlnge[i],dlnge_guinc[i]);
    fprintf(filedos,"\n");
    fflush(filedos);

    if(0==slowbin) brepeat=1;
}
while(1==1);

    t = time(NULL);
    local = localtime(&t);
    fprintf(filedos,"Time : %s", asctime(local));
    fprintf(filedos,"DOS estimated with increment unit of %.8lf\n",GAMMA);

    i=cinofbin;
    for(;i--;
    
    fprintf(filedos,"Density of state[%d] = %.3lf\n",i,dlnge[i]);
    fprintf(filedos,"\n");
    fflush(filedos);
    // Printing DOS after shifting
    i=cinofbin;
    for(;i--;
    
    dlnge[i]=dlnge[i]-dlnge[0];
    fprintf(filedos,"DOS after shifting [%d] = %.3lf\n",i,dlnge[i]);
    
    fprintf(filedos,"\n");
    fflush(filedos);
    
    fprintf("Done with GAMMA=%.8lf Running with
    GAMMA=%.8lf\n",GAMMA,GAMMA/3.0);
    GAMMA=GAMMA/3.0;
    }
    while(GAMMA-0.00000001>=0);

    fprintf(filedos,"\nFinal Density of States\n");
    i=cinofbin;
    for(;i--;
    
    fprintf(filedos,"Density of state[%d] = %.3lf\n",i,dlnge[i]);
    fprintf(filedos,"\n");
    fflush(filedos);

    t = time(NULL);
    local = localtime(&t);
    fprintf(filedos,"Finish Time : %s", asctime(local));
    printf("Program done\n");

    fclose(filecoorpointer);
    fclose(filecrystal);
    fclose(filedos);
    return 1;
    }