University of California

Los Angeles

Molecular Dynamics Simulation of Interfacial Tension and Contact Angle of Lennard-Jones Fluid

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mechanical Engineering

by

Shashank Sinha

2004

© Copyright by

Shashank Sinha

2004

The dissertation of Shashank Sinha is approved.

Gerassimos Orkoulas

Yongho "Sungtaek" Ju

Chih-Ming Ho

Jonathan B. Freund, Committee Co-chair

Vijay K. Dhir, Committee Co-chair

University of California, Los Angeles

2004

Dedicated to my mother,

Smt. Mira Sinha

and my father,

Prof. Arun K. Sinha.

Table of Contents

List of Figures . . . . **vii**

List of Tables . . . . **xi**

Nomenclature . . . . **xii**

Acknowledgment . . . . **xvii**

Vita . . . . **xviii**

Abstract of the Dissertation . . . . **xviii**

1 INTRODUCTION **1**

1.1 Literature Review 2

1.1.1 Surface Tension 2

1.1.2 Computer Experiments: Molecular Simulations 7

1.1.3 Lennard Jones Fluids: MD Study 7

1.1.4 Solid-Fluid interaction: MD study 11

1.1.5 Efficient and Accurate Force Evaluation 15

1.2 Objective of the Present Work 17

2 THEORY AND APPROACH **18**

2.1 Molecular Dynamics Simulation 18

2.2 Force Evaluation 20

2.2.1 Particle-Particle with Finite Cutoff 20

2.2.2 Verlet Neighbor List 20

2.2.3 Particle-Particle/Particle-Mesh (P^3M) Method 21

2.3 Modeling of the System 21

2.3.1 Solid-Fluid Interaction 21

2.4 Sampling 28

2.4.1 Density 28

2.4.2 Temperature 30

2.4.3 Pressure 31

2.4.4 Surface Tension 31

2.5 Hamaker Constant 35

3 SURFACE TENSION OF LENNARD JONES FLUID 37

3.1 Liquid-Vapor Equilibrium without External Field 37

3.2 Liquid-Vapor Equilibrium Adjacent to A Solid Wall 41

4 CONTACT ANGLE OF LIQUID DROPLET 55

4.1 Contact Angle 55

4.2 Droplet Simulation 56

5 IMPLEMENTATION OF P^3M METHOD FOR FORCES AND SURFACE TENSION 78

5.1 P^3M Method 78

5.2 The Short Range Term 79

5.3 The Long Range Term: Mesh Term 80

5.3.1 Charge Assignment 81

5.3.2 Potential Solver 83

5.3.3 Mesh Defined Field 84

5.3.4 Force Interpolation 85

5.4 Optimized Influence Function 86

5.5 Force Split for $1/r^p$ Type Term (Essmann *et al.* [1]) 87

5.6 Evaluation 91

5.6.1 Force Evaluation 91

5.6.2 Surface Tension Evaluation 92

5.7 Validation of Untruncated Force Evaluation 95

5.8 Validation of Untruncated Surface Tension Evaluation 104

5.9 Simulation of Liquid Film Using P^{3}M Method 112

6 CONCLUSIONS 116

6.1 Future Works 117

Appendix A: Code for Force and Surface Tension Evaluation using P^{3}M Method 118

Appendix B: Code for Droplet Simulation 173

References 188

List of Figures

1.1 Contact angle θ. 2

1.2 Schematic drawing of stresses near the interface. 3

2.1 FCC unit cell. A and B are two diagonally opposite atoms. 23

2.2 Schematic picture of the simulation domain to study the thickness effect on the surface tension. 27

2.3 Schematic picture of the simulation domain to study contact angle. 29

2.4 Interaction between a atom and solid surface. 36

3.1 Effect of cutoff radius on the density profile at various temperatures, (a) $T^* = 0.70$ and (b) $T^* = 0.85$ 39

3.2 Comparison of (a) the liquid density and (b) the vapor density with literature data as a function of temperature. 40

3.3 The surface tension of the liquid-vapor interface in a liquid slab geometry. 41

3.4 At $T^* = 0.85$, (a) local surface tension profile (b) density profile and the integrated surface tension profile. 43

3.5 For thick film ($\delta_f \approx 15\sigma$) (a) comparison of density (b) comparison of the surface tension. 45

3.6 Top view of the liquid film, domain cross sectional surface area is (a) $14 \times 14\sigma^2$ (b) $18.08 \times 18.08\sigma^2$. $T^* = 0.85$ 46

3.7 Variation of (a) the liquid density and (b) the vapor density with the film thickness, δ_f. $\epsilon_{sf} = 0.49\epsilon$, $\sigma_{sf} = 0.8\sigma$. 48

3.8 Variation of the surface tension with film thickness, δ_f. $\epsilon_{sf} = 0.49\epsilon$, $\sigma_{sf} = 0.8\sigma$. 49

3.9 Variation of (a) the liquid density and (b) the vapor density with the film thickness, δ_f. $\sigma_{sf} = 0.863436\sigma$. 52

3.10 Variation of the density of the first liquid layer adjacent to the solid surface with δ_f. $\sigma_{sf} = 0.863436\sigma$. 53

3.11 Variation of surface tension with the film thickness, δ_f. $\sigma_{sf} = 0.863436\sigma$. 54

4.1 Initial configuration for the liquid droplet simulation 58

4.2 Liquid droplet and the contact angle. 59

4.3 Droplet contours at $T^* = 0.7$. (a) $\epsilon_r = 0.408$, (b) $\epsilon_r = 0.572$ and (c) $\epsilon_r = 0.817$. Left one is center xz plane and right one is center yz plane. 61

4.4 Droplet contour at $T^* = 1.0$. (a) $\epsilon_r = 0.408$, (b) $\epsilon_r = 0.490$ and (c) $\epsilon_r = 0.572$. Left one is center xz plane and right one is center yz plane. 62

4.5 Variation of (a) the contact angle and (b) $\cos(\theta)$ with ϵ_r, ($\sigma_{sf} = 0.8\sigma$). 64

4.6 Variation of the contact angle with temperature ($\sigma_{sf} = 0.8\sigma$) . . . 65

4.7 Variation of (a) α and (b) β with temperature. (From Sullivan [2]) 66

4.8 Contact angle variation with temperature for (a) the organic fluids (From Adamson [3]) (b) the water (From Lay and Dhir [4]) 67

4.9 Variation of (a) the contact angle and (b) $\cos(\theta)$ with ϵ_r, ($\sigma_{sf} = 0.863436\sigma$). 68

4.10 Variation of the contact angle with temperature ($\sigma_{sf} = 0.863436\sigma$). 69

4.11 Variation of $\gamma_{sv} - \gamma_{sl}$ with temperature for (a) $\sigma_{sf} = 0.80\sigma$ and (b) $\sigma_{sf} = 0.863436\sigma$). 70

4.12 Variation of the contact angle with temperature for different ψ_{min} 72

4.13 Variation of (a) α and (b) β with temperature. 75

4.14 Comparison of the contact angle predicted from the correlation and simulation data's. 76

4.15 Contact angle dependence on (a) Hamaker constant, A and (b) $A/(\rho_{liq}\sigma_{sf}^3)$. 77

5.1 Comparison of LJ force using P^{3}M method with exact calculation 96

5.2 Magnified view of Fig. 5.1 at (a) $r = 1.8\sigma \sim 3.1\sigma$ and (b) $r = 5.6\sigma \sim 5.7\sigma$ 98

5.3 Absolute difference of Force$_{\text{exact}}$ and Force$_{\text{P}^3\text{M}}$ for mesh points $= 32 \times 32 \times 32$, and $r_{cutoff}^{sr} = 3.0\sigma$. 100

5.4 Variation of (a) the short range force and (b) the long range force with β. $N_c = 32^3$ 101

5.5 Variation of error in force with mesh points N_c and β. 102

5.6 Variation of error in force with β for different cutoff radius. 103

5.7 Comparison of γ term using P^{3}M method with exact calculation, $N_c = 48^3$, $\beta = 1.0$ 105

5.8 Magnified view of Fig. 5.7 at (a) $r = 1.6\sigma \sim 3.1\sigma$ and (b) $r = 2.8\sigma \sim 6.0\sigma$ 107

5.9 Absolute difference between γ_{exact} and $\gamma_{\text{P}^3\text{M}}$ for $r^{sr}_{cutoff} = 3.0\sigma$. (a) β is verifying, $N_c = 48^3$ and (b) N_c is varying, $\beta = 1.0$ 108

5.10 Variation of (a) γ_{sr} and (b) γ_{lr} with r. $N_c = 48^3$ 109

5.11 Variation of error in surface tension term with mesh points N_c and β. 110

5.12 Variation of error in γ with β for different cutoff radius. 111

5.13 Density profiles at various temperatures 113

5.14 Comparing the liquid and the vapor density with the density of Argon fluid in thermodynamic equilibrium 114

5.15 Comparison of the surface tension 115

List of Tables

3.1 Simulation parameters 37

3.2 Densities and the surface tension of thin films, $\sigma_{sf} = 0.80\sigma$ 47

3.3 Densities and surface tension of thin films, $\sigma_{sf} = 0.863\sigma$ 50

4.1 Value of ψ_{min} used in the simulations. 71

5.1 Function f_p and g_p in Eq. (5.39) and Eq. (5.40) 90

Nomenclature

$A, \mathcal{A}$	Surface Area
E	Total energy of the system
E_k	Kinetic energy
G	Green function
H	Mesh cell length
LJ	Lennard-Jones
L_x, L_y, L_z	Length of the simulation box along x, y and z axes respectively
N	Normal to the interface
N, N_{atm}	Number of Atoms
N_f	Number of degrees of freedom
N_{bin}	Number of bins
P	Pressure
R	Radius of droplet, Universal gas constant
R_0	Lattice constant
T	Tangent to the interface
T	Temperature of the system
TS	Truncated and shifted

V	Volume of the system
Δt	Time Step
Γ	Gamma function
Σ	Stress, Sum
α, β	Different phases
δ	Tolman length, Kronecker delta function
δ_f	Film thickness
ϵ	Lennard-Jones 12, 6 energy parameter
ϵ_r	Relative strength of the wall, $\epsilon_{sf}/\epsilon_{ff}$
γ	Surface Tension
γ_{lv}	Surface Tension of liquid-vapor interface
γ_{sl}	Surface Tension of solid-liquid interface
γ_{sv}	Surface Tension of solid-vapor interface
$\hat{}$	Fourier transform
∞	Infinity
$\langle \ldots \rangle$	Ensemble average
$\mathbf{F}$	Force
$\mathbf{v}$	Velocity
ϕ	Potential function

ψ	Wall potential
ρ	Density
$\rho^{(1)}$	Singlet density
$\rho^{(2)}$	Doublet or pair density
σ	Length parameter in LJ (12, 6) potential
θ	Contact Angle
ξ_{sl}	$\gamma_{sl} - \gamma_s$
ξ_{sv}	$\gamma_{sv} - \gamma_s$
$'$	Derivative
*	Non-dimensional parameters
c	Critical point
d	Interfacial thickness
dv	Volume element
g	Radial distribution function
i	Imaginary quatity, $\sqrt{-1}$
i, j	Atoms index
k, k_B	Boltzman constant
l	Length
l	Liquid

lr	Long range
m	Mass of atom
min	Minimum
n	Time index
new	New step
old	Old step
q	Charge
r	Separation distance between atom
r_c	Cutoff radius
r_l	Neighbor list radius
s	Solid
sr	Short range
t	Triple point
$target$	Target quantity
v	Vapor
x	Along x dirextion
y	Along y dirextion
z	Along z dirextion
z_e	Equi-molar dividing surface

z_s	Surface of tension
$\mathbf{r_{12}}$	Relative position of 1 w.r.t 2
$\mathbf{r}$	Position
$\mathcal{M}$	Molecular weight

Acknowledgments

I would like to thank Professor Dhir and Professor Freund for their enthusiastic guidance in the research work and providing the necessary financial support through NASA microgravity research program. I would like to thank Professor Eric Darve (Stanford University) for helping in the implementation of P^3M method. I am also greatful to Professor Orkoulas, Ho and Ju for their valuable suggestions.

I would like to thank my parents for encouraging and guiding me in the right path since the childhood. I would like to thank my sisters, brother-in-laws, brother, nephew and niece for their moral support and inspiration during my graduate study which kept me motivated.

Finally, I would like to thank all my friends and relatives for their help. I am grateful to colleagues at the Boiling Heat Transfer Laboratory for their helpful suggestions and support.

Vita

1974	Born, Gaya, India
1994–1998	B.Tech. (Mechanical Engineering), Indian Institute of Technology, Kanpur, India
Summer 1997	Summer Intern, Engine Division at Tata Engineering and Locomotive Company (TELCO), Jamshedpur
1998–2004	Graduate Student Researcher, Mechanical and Aerospace Engineering Department, UCLA
1999	M.S. (Mechanical Engineering), UCLA
Fall 2000	Teaching Associate, Mechanical and Aerospace Engineering Department, UCLA. Taught discussion section of Intermediate Heat Transfer course under the direction of Prof. Adrienne Lavine

Publications and Presentations

Sinha, S.; Dhir, V. K. and Freund, J. B. (November, 2000). Atomistic Simulation of the Surface Tension of Ultra Thin Liquid Films as Influenced by Solid Wall, Presented at APS meeting (Division of Fluid Dynamics), session DP, Washington D.C.

Sinha, S.; Freund J. B. and Dhir V. K. (November, 2001). Molecular Dynamics Simulation of Interfacial Tension of Ultra-Thin Liquid Films Formed on a Solid Surface, Proceeding of 2001 International Mechanical Engineering Congress and Exposition (Winter Annual Meeting), New York, NY

Sinha, S.; Freund, J. B.; Dhir, V. K.; Darve, E. and Shi, B. (July, 2003). Surface Tension Evaluation in Lennard-Jones Fluid System with Untruncated Potentials, Proceeding of ASME Summer Heat Transfer Conference, Las Vegas, NV.

Abstract of the Dissertation

Molecular Dynamics Simulation of Interfacial Tension and Contact Angle of Lennard-Jones Fluid

by

Shashank Sinha

Doctor of Philosophy in Mechanical Engineering

University of California, Los Angeles, 2004

Professor Vijay K. Dhir, Co-chair

Professor Jonathan B. Freund, Co-chair

Molecular techniques have been used to study interfacial tension for more than half century. Interfacial tensions regulate various phase change phenomena and heat transfer, especially when phase change occurs. At the sub-micron scales of MEMS devices, microscale evaporation and condensation, surface effects can dominate. The dynamics of thin film is studied here to quantify the effect of film thickness, system temperature and wall strength. When the film is thick, liquid-vapor interfacial tension is evaluated by integrating the difference between normal and tangential components of pressure tensor across the interface. Interfacial tension dependence on the film thickness is investigated and found to be weakly dependent. Strength of the solid surface plays an important role for a stable film adjacent to the solid surface.

Liquid droplet is simulated adjacent to a semi-infinite solid surface. Contact angle of a static droplet is investigated at different temperatures and the solid-fluid interaction strength. Hamaker constant of the fluid-solid combination, fluid density and solid-fluid Lennard-Jones length parameter are found to be important parameters controlling the contact angle. Variation of the contact angle with all necessary parameters are discussed.

Interfacial tension of the liquid-vapor interface is calculated using Molecular Dynamics simulation with tail correction to correct the finite cutoff radius used in the simulation. The resultant surface tension, liquid density and vapor density are found to be well predicted when compared with experimental data for Ar (LJ fluid). Liquid and vapor densities were found to depend on the finite cutoff radius which motivates the use of an untruncated force/potential calculation using P^3M (particle-particle particle-mesh) method which was implemented for force and surface tension evaluation. Each term is computed by splitting it into short and long range parts. This does not require the tail correction. It is found to be very accurate as well as promise to be computationally efficient for larger system. In our case, it is found to be similar to computational time for neighbor-list method with cutoff radius of 4.5σ.

CHAPTER 1

INTRODUCTION

Due to their fundamental importance for many technological processes, interfacial properties have been studied extensively in several experimental and theoretical investigation. They control phase change phenomena (boiling, condensation etc.), wetting and drying of the solid wall, adhesion and stiction between mechanical components, capillarity effect in narrow slits. However experimental study of the interfacial properties is difficult and challenging. In the last decade, computer simulations have complemented our understanding of the properties of pure homogeneous fluids and mixture as well as on their interfacial behavior.

An interface acts as a stretched elastic membrane. The membrane tension is the surface tension. When a small drop is placed close to a solid surface, interfacial tensions between solid, liquid and vapor phase give it a definite shape and contact angle, θ between solid surface and liquid-vapor surface from liquid side as shown in Fig. 1.1. The line at which all three phases meet is the contact line. Contact angle $\theta = 0°$ is a perfectly wetting case and $\theta = 180°$ is a perfect dry case. For a continuum, it is related to the interfacial tensions through Young's equation,

$$\gamma_{sv} = \gamma_{sl} + \gamma_{lv} \cos\theta. \tag{1.1}$$

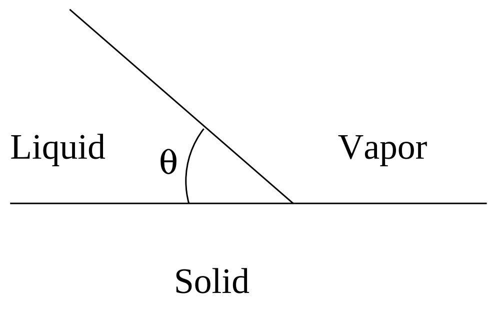


Figure 1.1: Contact angle θ.

1.1 Literature Review

1.1.1 Surface Tension

According to the thermodynamic definition, surface tension is defined as the isothermal work of formation of an unit area of interface. Using statistical mechanics, Fowler [5] developed a relation between the surface tension and the intermolecular forces. However, as in earlier work of others, Fowler introduces the approximation of a mathematical surface of density discontinuity between the two phases. By neglecting the density of the vapor and calculating the work done when the liquid is brought into two semi-infinite halves by an isothermal procedure, he obtained the following expression for the surface tension

$$\gamma = \frac{\pi}{8}\rho_l^2 \int_0^\infty r^4 \frac{d\phi(r)}{dr} g(r) dr, \tag{1.2}$$

where ρ_l is the liquid density, $g(r)$ is the radial distribution function and ϕ is the interatomic potential. At lower temperatures, the approximation of density discontinuity can be justified, but it can produce a large error near the critical point. The first attempt to express the surface tension in terms of the intermolecular potential and the distribution functions near the transition layers was made

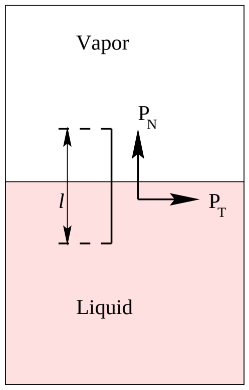


Figure 1.2: Schematic drawing of stresses near the interface.

by Kirkwood and Buff [6]. They calculated the stress tensor in the region near the transition layer and obtained the expression for the principal stresses P_T and P_N. According to their theory, the normal stress P_N, which is equal to the vapor pressure of the substance, is given by,

$$\begin{aligned} P_N = P \quad &= kT\rho_\alpha^{(1)}(\mathbf{r_1}) - \tfrac{1}{6}\int\int\int r_{12}\tfrac{d\phi}{dr}\rho_\alpha^{(2)}(\mathbf{r_1},\mathbf{r_{12}})dv_{12} \\ &= kT\rho_\beta^{(1)}(\mathbf{r_1}) - \tfrac{1}{6}\int\int\int r_{12}\tfrac{d\phi}{dr}\rho_\beta^{(2)}(\mathbf{r_1},\mathbf{r_{12}})dv_{12}, \end{aligned} \tag{1.3}$$

where α and β are fluid phases, $\rho^{(1)}(\mathbf{r})$ specifies the average number of molecules $\rho^{(1)}dv$ in a volume element dv at a point $\mathbf{r}$ in the fluid and $\rho^{(2)}(\mathbf{r_1},\mathbf{r_{12}})$ specifies the average number of molecular pairs $\rho^{(2)}dv_1dv_{12}$, one member of which is situated in volume element dv_1 at point $\mathbf{r_1}$ and the other in volume element dv_{12} at point $\mathbf{r_{12}}$ in the relative configuration space of the pair. The pair correlation function $g^{(2)}(\mathbf{r_1},\mathbf{r_{12}})$ is defined by the relation,

$$\rho^{(2)}(\mathbf{r_1},\mathbf{r_{12}}) = \rho^{(1)}(\mathbf{r_1})\rho^{(1)}(\mathbf{r_2})g^{(2)}(\mathbf{r_1},\mathbf{r_{12}}). \tag{1.4}$$

Tangential stress components are calculated as the sum of the momentum transport and the force transmitted across a strip of unit width and length l perpendicular to the transition layer. Kirkwood and Buff expressed this stress Σ_x in terms of the intermolecular forces and the pair distribution function to obtain

$$\Sigma_x = -\int_{-l/2}^{l/2} P_T(z)dz, \tag{1.5}$$

where

$$P_T(z) = P^{'} = kT\rho^{(1)}(z) - \frac{1}{2}\int\int\int \frac{x_{12}^2}{r_{12}}\frac{d\phi}{dr_{12}}\rho^{(2)}(z, r_{12})dv_{12}. \tag{1.6}$$

The stress acting in the direction across such a strip of length l perpendicular to the interfacial plane, would have the value $-Pl$ resulting from the uniform normal pressure P acting in both phases α and β, if there is no interfacial contribution. The tension γ, in excess of the contribution $-Pl$ as a result of the uniform normal pressure, acting across the strip is, according to the mechanical definition, the interfacial tension of the phase boundary. Thus we can write,

$$\gamma = \Sigma_x + \int_{-l/2}^{l/2} Pdz, \int_{-l/2}^{l/2} Pdz = Pl. \tag{1.7}$$

This has became widely accepted and provides a means to compute the surface tension at the molecular scale. Substitution of Eq. (1.5) in Eq. (1.7) and in the limit $l \to \infty$, yields for the surface tension

$$\gamma = \int_{-\infty}^{\infty} [P_N - P_T]dz. \tag{1.8}$$

Harasima [7] critically examined the above theory and studied various aspects of the surface tension using thermodynamics. He related the surface energy to the heat of vaporization and critically evaluated approximate theories of surface tension used by others to facilitate computation. He also discussed quantum-mechanical treatments and related Toda's quantum-mechanical formula to the classical one by using the density matrices. However mechanical definition by Kirkwood and Buff, has been used for several decades due to its clear physical interpretation and a ever growing computer power.

Tolman [8] addressed the dependence of surface tension on the curvature of the interface using thermodynamic arguments. He showed that the surface tension of a liquid drop depends on radius R as

$$\gamma(R) = \gamma_\infty(1 - \frac{2\delta}{R}) + O(R^{-2}), \tag{1.9}$$

when γ_∞ is the surface tension of a planner liquid-vapor interface. He expressed the Tolman length δ in terms of the difference between the equi-molar dividing surface of a planar interface z_e and the so-called surface of tension z_s,

$$\delta = z_e - z_s. \tag{1.10}$$

As both dividing surfaces lie in the interfacial region, the Tolman length is expected to be of molecular size, which means that only at the molecular scales does curvature alter the surface tension.

Droplets of this scales are difficult to observe experimentally; estimates of δ are obtained from theoretical models and simulation. Haye and Bruin [9] simulated a three dimensional slab of Lennard Jones (LJ) Fluid in equilibrium with its vapor on both sides. They used 12-6 truncated and shifted potential with cutoff radius of 2.5σ:

$$\phi_{TS}(r) = \begin{cases} \phi_{LJ}(r) - \phi_{LJ}(2.5\sigma), & \text{if } r \leq 2.5\sigma \\ 0, & \text{if } r > 2.5\sigma \end{cases}, \tag{1.11}$$

where

$$\phi_{LJ}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]. \tag{1.12}$$

Eq. (1.12) is known as Lennard-Jones potential with ϵ and σ as energy and length parameters respectively. They simulated at six different temperatures ranging from the triple-point temperature to the critical-point temperature. For all temperatures, the Tolman length was obtained from the MD simulation. The Tolman length was found to be a small positive number $\approx 0.2\sigma$, indicating that the surface tension of a droplet decreases slowly with decreasing drop size. Neijmeijer *et al.* [10] looked at the surface tension of a small droplet and tried to estimate the Tolman length δ, by evaluating the pressure difference across a liquid droplet boundary which leads to an estimate of Tolman's length. They found it to be small and positive.

Pressure stress tensor for plannar surface was described by Nijmeijer *et al.* [11]. He analyzed two different definitions of tangential stress components that give the same value of the surface tension but lead to different values for the apparent height at which the tension acts.

Several researchers have examined the pressure tensor of spherical interfaces. Blokhuis and Bedeaux [12] used an expression for the pair density and found the location of the mechanical surface of tension and Tolman's length which shows good agreement with the values found by Nijmeijer [11]. However, they argued that the distance between the so-called mechanical surface of tension and the Gibbs dividing surface is not given by Tolman's length. Bardouni *et al.* [13] studied the local pressure in spherical liquid-vapor interfaces using MD simulations. They found that the surface tension of the interface is independent of the curvature for the curvatures investigated. However investigation of larger curvatures (small radii) was not possible because of considerable statistical noise encountered on the liquid side of the interface.

1.1.2 Computer Experiments: Molecular Simulations

Molecular simulation is a statistical mechanics method to obtain a set of configurations distributed according to some statistical distribution function, or statistical ensemble. There are two primary types of molecular simulation: Molecular Dynamics (MD) and Monte Carlo (MC). In MD, equations of motion are integrated to track atoms. It is a deterministic technique: given an initial set of positions and velocities, the subsequent time evaluation is completely determined, whereas MC is a statistical method to sample phase-space faster by moving the atoms randomly and system properties can be statistically obtained. Several books describe the methods in detail: Allen and Tildesley [14], Frenkel and Smith [15], Rappaport [16].

1.1.3 Lennard Jones Fluids: MD Study

Molecular Dynamics and Monte Carlo simulations have both been used extensively to study the liquid-vapor equilibrium for model Lennard-Jones (12, 6) fluids. Though crude, this model provides a reasonably accurate description of liquid-vapor coexistence. Its ease of implementation makes it the best studied model.

Chapela *et al.* [17] simulated LJ atoms using MC and MD methods at different temperatures to study the surface tension. He used 255, 1020 and 4080 LJ atoms and cutoff radius of 2.5σ. Molecules were placed in a doubly periodic (in x &

y direction) box, whose z boundary was composed of molecularly homogeneous substance (one side density is the same as liquid and other side is the same as vapor). So, a well defined liquid-vapor interface forms in the middle of the simulation box. He concluded that for 255 atoms, both MC and MD give the same result. Density profiles were well fitted by a hyperbolic tangent. The thickness of the interface was found to be of the order of two molecular diameters and increased rapidly as the critical point is approached. They developed a tail correction for surface tension to account for the finite cutoff radius used in the simulation. Later on, Blokhius *et al.* [18] corrected this correction expression.

Nijmeijer *et al.* [11] simulated a liquid slab of LJ atoms in equilibrium with its vapor on both sides. They used up to 10,000 atoms at four different non-dimensional temperatures T^* $(= \epsilon/k_B)$ of 0.72, 0.80, 0.90 and 1.00. Initial configurations were taken from earlier simulations. They accumulated statistics over 12,000 time-steps after an equilibration period of 4,000 time-steps. A non-dimensional time-step $\Delta t^*(= \Delta t\sqrt{\epsilon}/\sigma\sqrt{m})$ of 0.01 was used in their simulation. To see the effect of the cutoff radius, they also simulated a system with much larger cutoff radius (7.33σ) for $T^* = 0.92$ and found that the surface tension increases by a factor of 2.8. It was concluded that the discrepancy can be ascribed partly to the ommittance of the attractive tail of the truncated potential.

The slab geometry has been studied by many researchers in order to investigate the simulation setup and the simulation parameters to obtain reliable data. Holcomb *et al.* [19] and Meche *et al.* [20] critically investigated the appropriate simulation parameters. They concluded that a cutoff radius of 5.0σ supplemented by a tail correction is required to obtain a reliable value for the surface tension. Meche *et al.* [21] also studied structure and equilibrium properties of the liquid-vapor interfaces of binary mixtures containing argon and methane over the entire range of compositions. From both the surface tension results and the partial density profiles the relative enrichment of argon at the interface was estimated.

Despite considerable research there does not appear to be a consensus concerning coexistence properties in the literature. Trokhymchuk and Alejandre [22] studied liquid-vapor equilibrium using both MD and MC simulations. They performed the simulations with 1000 LJ atoms in a parallelepiped box. Although different surface tension values are attributed in literature to different setup conditions such as size of simulation cell, number of particles, cutoff radius and time of simulation. They argued that the origin of the discrepancy lies in the details of truncation procedure used. They noted that a truncated potential in MC is not equivalent to truncated force used in MD. The truncated force does not uniquely define the primordial potential. Simulations were used to support their arguments. They also simulated the system with higher cutoff radii (4.5σ and 5.5σ) and found that surface tension calculated from the simulation is more sensitive to the potential tail. For a truncated potential, γ increases 35% and 5% when cutoff radius increases from 2.5σ to 4.5σ and from 4.5σ to 5.5σ, respectively. When the potential is truncated and shifted, the same cutoff change γ by more than 60% and more than 10%, respectively.

Chen [23] studied the area dependence of the surface tension for LJ fluid. It was found that surface tension increases with the decrease of the cross sectional surface area. However, this finite size was pronounced only in small surface area. In addition, their simulation showed that a finite size correction of the surface tension is directly proportional to the reciprocal of the surface area. Weng *et al.* [24] used MD simulation to study a Lennard-Jones liquid thin film suspended in its vapor and explored the film thickness effect on its stability. Their simulation results indicated that the local surface tension distribution varied significantly with film thickness, while surface tension values and density profile shows little variation. As the film gets thinner, the two liquid-vapor interfacial regions begin to overlap and liquid phase molecules in the center region of the film experience larger tension in the direction parallel to the film surface. Such interface overlapping is believed to destabilize the film and the occurrence of film rupture depends on the system temperature and the cross-sectional area of the computational domain.

Kawano [25] looked at the instability wave on a 1.06 nanometer diameter liquid thread using MD. He used up to 10278 LJ atoms in the simulation. The rupture phenomena in a liquid thread and the formation of ultra-fine liquid particles were successfully simulated for various conditions. The numerical results of the interfacial phenomenon in the liquid thread, which include the unstable wave motion and the rupture time, were quantitatively compared with theoretical results based on the classical linear instability theories and wavelength were found in reasonable agreement with those obtained using the inviscid linear instability theory. Moseler and Landman [26] studied unstability of nanojets with velocities up to 400 m/s using large scale molecular dynamics simulations. Nanojets are created by a simulated pressurized injection of fluid propane through nanoscale convergent gold nozzles with heating or coating of the nozzle exterior surface to prevent formation of thick blocking films. Emergence of double-cone neck shapes were predicted when the jet approaches nanoscale molecular dimension, deviating from the long thread universal similarity solution obtained in absence of such fluctuations.

There are few molecular simulations of binary/ternary mixtures of LJ atoms and studies of interfacial tension of mixture fluids (Herrera *et al.* [27], Stecki and Toxvaerd [28], Meyer *et al.* [29], Shiraichi and Hagiwara [30]). Binary mixtures are modeled by modifying the LJ potential function between dissimilar atoms as,

$$\phi_{ss'}(r) = 4\beta\epsilon_{ss'} \left[\left(\frac{\sigma_{ss'}}{r} \right)^{12} - \alpha_{ss'} \left(\frac{\sigma_{ss'}}{r} \right)^{6} \right], \tag{1.13}$$

and so the relative interaction strength between atom s and $s^{'}$ can be changed by changing β and $\alpha_{ss'}$ and partially miscible to immiscible mixtures can be simulated. Bresme and Quirke [31] used a similar potential to investigate the wetting behavior of nanoscale liquid lenses at a liquid-liquid interface. The spreading of the lens was controlled by changing the lens-liquid surface tension. They used up to 30,000 atoms for the surrounding and 3,000 atoms for the liquid lens. Measured contact angles agreed well with the macroscopic description provided by Neumann's equation. It was also found that Laplace's equation for the pressure difference between the interior of the lens and the liquid phases was obeyed.

Barker [32] calculated the surface tension of Ar, Kr and Xe interface with and without Axilroad-Teller-Muto three body potential using Monte Carlo simulations. He showed that the calculated results match well with the experimental result only when the three-body potential is used. However, other researchers have claimed that interfacial properties are predictable with pair potential. Fluid phase equilibria was studied for chlorine and hexane molecules (Alejandre *et al.* [33]) too. Cl_2 was modeled as a rigid diatomic molecule, and n-hexane as an isotropic united-atom model.

1.1.4 Solid-Fluid interaction: MD study

Solid-Fluid interactions are encountered in many practical applications. Fluid behavior near the solid surface determines its wetting and drying properties, film stability and contact angle. The solid-fluid interaction potential has been modeled in a number of ways in the past. In many cases, the simple Lennard-Jones (12,6) interaction is used with ϵ_{sf} and σ_{sf} determine the tendency for the surface to be wetted by the liquid. One can integrate the effect of all atoms (x and y direction) in a particular layer (FCC (111) plane) and get an equivalent potential (10, 4) as

$$\psi_s(z) = \frac{4\sqrt{3}\pi}{15}\frac{\epsilon_{sf}\sigma_{sf}^2}{R_0^2}\left[2\left(\frac{\sigma_{sf}}{z}\right)^{10} - 5\left(\frac{\sigma_{sf}}{z}\right)^4\right], \tag{1.14}$$

where R_0 is the lattice constant of the solid surface. From the Eq. (1.14) the minimum energy is obtained at $z = \sigma_{sf}$

$$\psi_{min} = -\frac{4\sqrt{3}\pi}{5}\left(\frac{\sigma_{sf}}{R_0}\right)^2 \epsilon_{sf}. \tag{1.15}$$

If we consider an infinite number of FCC(111) planes in $-z$ direction then we get an equivalent (9, 3) potential

$$\psi_s(z) = \frac{2\sqrt{2}\pi}{45}\frac{\epsilon_{sf}\sigma_{sf}^3}{R_0^3}\left[2\left(\frac{\sigma_{sf}}{z}\right)^{9} - 15\left(\frac{\sigma_{sf}}{z}\right)^3\right]. \tag{1.16}$$

Maruyama *et al.* [34] suggested that the combinations of σ_{sf} and ϵ_{sf} which give the same value for ψ_{min} always yield the same contact angle. Maruyama *et al.* [35] looked at the phase change of the liquid droplet in contact with the solid surface. Solid surfaces on the top and bottom of the calculation domain were represented by three layers of harmonic molecules with additional "phantom" molecules beyond this. The phantom molecules, which attempted to mimic the continuous bulk solid, were used to provide constant-temperature boundary conditions. Bottom and top surfaces were maintained at higher and lower temperatures so evaporation and condensation occurred. Velocity profiles, temperature distributions were calculated for evaporating and condensing droplets.

Matsumoto [36] investigated the molecular diffusion behavior near the contact line of solid, liquid and vapor phases. A comparatively large diffusion along the solid-liquid and the liquid-vapor interface was detected.

Bubble and droplet nucleation on solid surfaces were studied by Kimura and Maruyama [37]. Nakabe *et al.* [38] compared the micro-droplet behavior situated on a atomically flat and rugged surface. They concluded that the contact angle decreases with an increase in well depth Eq. (1.15) parameter of the argon-platinum potential function for both flat and rugged cases. Also, at a constant well depth parameter, the contact angle for the rough surface was found to be much larger than the one for the flat surface, which indicates that the micro-scale rugged surface structure could be effective for liquid-resistant surface treatment.

Wetting and drying of inert solid walls by LJ fluid has been simulated extensively. Saville [39] carried out MD simulations of LJ atoms in a rectangular box to calculate the contact angle using Eq. (1.1). The determination of γ_{sv} and γ_{sl} becomes very difficult and so (following Gibbs) he replaced it with ξ_{sv} and ξ_{sl} respectively defined by,

$$\xi_{sv} = \gamma_{sv} - \gamma_s, \tag{1.17}$$

and

$$\xi_{sl} = \gamma_{sl} - \gamma_s, \tag{1.18}$$

where γ_s is the surface tension of solid against vacuum, which converts Young's equation Eq. (1.1) to

$$\xi_{sv} = \xi_{sl} + \gamma_{lv} \cos\theta. \tag{1.19}$$

He calculated ξ as

$$\xi = \gamma - \gamma_s = \frac{1}{\mathcal{A}}\left[\left\langle \sum_{i<j}\sum_{j}\frac{x_{ij}^2 - z_{ij}^2}{r_{ij}}\phi^{'}(r_{ij})\right\rangle - \left\langle \sum_{i} z_i\psi^{'}(z_i)\right\rangle\right], \qquad (1.20)$$

where $\mathcal{A}$ is the surface area of the interface in the simulation domain. He used Young's equation to calculate contact angle at the triple point temperature though the validity of Young's equation in microscale remained to be subject of arguments. At the same time Navascués and Barry [40] developed expressions for the work of adhesion W_A, which was developed independently by Saville [39]. He found that W_A consists of two terms $W_A(1)$ and $W_A(2)$. The first term depends directly on the solid-fluid interaction potential and the fluid one-particle distribution function, and corresponds to the direct interaction between fluid and solid whereas the second term depends on the fluid-fluid interaction potential and the fluid two particle density distribution function, and corresponds to the relaxation of the fluid density profiles to its free surface form when the liquid is pulled away from the solid.

Sikkenk *et al.* [41] and Nijmeijer *et al.* [42] used the above formulation to study the wetting and drying of a FCC(100) solid wall by LJ atoms. Simulations consisted of two types of molecules: one for three layer substrate and the second composing the fluid. Fluid atoms were placed between two solid walls and simulations were performed at constant temperature and variable ratio of substrate-adsorbate to adsorbate-adsorbate attraction. They observed three phenomena depending on the value of $\epsilon_r = \epsilon_{sf}/\epsilon_{ff}$.

1. For $\epsilon_r < 0.54$, the liquid layer resides in the middle of the system and both sides are covered with vapor. This case has two solid-vapor and two liquid-vapor interfaces. The walls are dry.

2. For $0.54 < \epsilon_r < 0.78$, the liquid is adsorbed at one side wall, the other is dry. This case has one each of solid-liquid, liquid-vapor and solid-vapor interface.

3. For $\epsilon_r > 0.78$, both sides of the walls are covered with liquid. This case has two solid-liquid and two liquid-vapor interfaces. The walls are fully wet.

Tang and Harris [43] investigated the fluid wetting phenomena on molecularly rough surfaces as they correspond to some physical situations where there is surface irregularity, such as surface defects due to vacancies or surface structure caused by adsorption of foreign particles onto a perfect surface. They used three layers of solid particles arranged in a FCC lattice for smooth surface. A molecularly rough surface was represented by adding to the smooth surface a fourth incomplete layer. They found that density profiles near the rough surfaces are different from those near the smooth surface and reflect the detailed molecular structure of the surfaces and the contact angle was found to be altered due to this surface irregularity. It was also concluded that the wetting transition on molecularly rough surfaces occurs at a higher value of the liquid-solid interaction strength than for a smooth surface.

1.1.5 Efficient and Accurate Force Evaluation

The accurate and efficient calculation of inter-particle forces/energies and system properties present a formidable challenge in the computer simulation. The famous *Ewald sum* (Ewald [44], de Leeuw *et al.* [45, 46]) for the coulomb potential (as in case of water) does a remarkable job by splitting the potential into two sums which converge very fast and avoids large error which may appear due to the finite cutoff for coulomb potential (Deserno and Holm [47]). The use of FFT, accelerate the convergence and several methods have been developed, the so called particle-particle/particle-mesh (P^{3}M) (Hockney and Eastwood (1988)), particle mesh Ewald (PME) (Darden *et al.* [48]) and smooth PME methods (Essmann *et al.* [1]). This method has been used for the Coulomb potential, which is slowly decaying $\sim r^{-1}$.

In the case of LJ atoms and in the case of the r^{-6} dispersion interaction in particular, most have used a cutoff followed by correction based on a uniform mean density beyond the cutoff. However interfacial properties of LJ fluids are shown very sensitive to the finite cutoff used in the simulation (Nijmeijer *et al.* [11]) and that motivates to use an accurate potential evaluation for LJ atoms.

Though uncommon in literature, mesh based methods like (P^{3}M) or PME can be used to evaluate an untruncated r^{-p} potential (Essmann *et al.* [1]). In the P^{3}M method, the force is split into long range and short range parts

$$\mathbf{F}_{ij} = \mathbf{F}_{ij}^{sr} + \mathbf{F}_{ij}^{lr}, \tag{1.21}$$

where $\mathbf{F}_{ij}^{sr}$ has effective compact support for only a few interparticle distances and $\mathbf{F}_{ij}^{lr}$ is sufficiently smooth to be accurately represented on a mesh. A neighbor list method is used to evaluate $\mathbf{F}_{ij}^{sr}$ with $\mathcal{O}(N)$ operations using the particle-particle algorithm. A discrete Poisson equation solver is used to compute $\mathbf{F}_{ij}^{lr}$ in $\mathcal{O}(N \log N)$ operations. The Green's function in the Poisson solver is selected to minimize the net error of the overall numerical method. Essmann *et al.* [1] has formulated Ewald sum like expressions for potentials of the form $1/r^p$ with $p \geq 1$ which we incorporate into a P^{3}M algorithm for r^{-6} and to compute surface tension.

1.2 Objective of the Present Work

There are three objectives of the present study:

1. Examine the surface tension of a very thin liquid film: The surface tensions of liquid-vapor and solid-vapor interfaces have been evaluated but their relationship was not studied. There is also no work reported in the literature which shows the effect of wall potential or film thickness on interfacial tension. We investigated the surface tension of a liquid film placed on a solid surface as its dependence on the film thickness and temperature.

2. Contact angle of liquid droplet: Contact angle controls the bubble dynamics and heat transfer. Droplets are simulated with few layers of solid atoms in literature, but these simulations did not include a semi-infinite solid. A semi-infinite solid produces a realistic potential field for practical applications. A systematic study of contact angle dependence on various parameters is investigated

3. Implementation of $\mathrm{P^3M}$ method for force and surface tension evaluation: Interfacial properties are found to be very sensitive to the finite cutoff radius. We implemented $\mathrm{P^3M}$ method for the dispersion force term (r^{-6}) while truncated the short range repulsive term at a few atomic thickness. Similar approach is applied for the surface tension evaluation and does not include any tail correction. This can be used for various molecular simulations to avoid any truncation.

CHAPTER 2

THEORY AND APPROACH

This chapter details the Molecular Dynamics simulation method to be used to study the dynamics of thin liquid layers. The following section provides an overview of the method and subsequent sections detail methods for evaluating interatomic forces, modeling the system and computing statistics.

2.1 Molecular Dynamics Simulation

Molecular Dynamics (MD) simulation is used to determine the properties of a classical many-body system, where atoms and molecules follow trajectories based on Newton's Law. The full system consists of atoms, modeled this way plus boundaries and external potentials. In classical molecular dynamics simulation, atomic positions evolve from their initial condition $(\mathbf{r}_0, \mathbf{v}_0)$ according to Newton's equation of motion

$$\frac{d^2\mathbf{r}_i}{dt^2} = \frac{\mathbf{F}(\mathbf{r_i})}{m_i} \tag{2.1}$$

where $\mathbf{r}_i$ is the position of atom i, m_i is its mass and $\mathbf{v}_i$ is its velocity. Except for trivial situations, Eq. (2.1) must be solved numerically. We use the *Velocity Verlet Algorithm*,

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \mathbf{v}_i^n \Delta t + \frac{\mathbf{F}_i^n}{m_i}\frac{(\Delta t)^2}{2}, \tag{2.2}$$

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \frac{(\mathbf{F}_i^n + \mathbf{F}_i^{n+1})}{m_i}\frac{\Delta t}{2}, \tag{2.3}$$

which is 2^{nd} order accurate in time. Subscript n indicates a quantity at time $t = n\Delta t$ where Δt is the numerical timestep.

Systems of LJ atoms are typically non-denationalized using k_B, σ, ϵ and m. This gives

- Time : $\sigma\sqrt{m/\sigma}$
- Temperature : ϵ/k_B
- Surface Tension : ϵ/σ^2
- Pressure : ϵ/σ^3
- Number Density : σ^{-3}.

The simulation starts with an initial configuration of atoms and initial velocities are assigned to the atoms randomly and scales so that

$$\frac{1}{N_{atm}}\sum_{i=1}^{N_{atm}}\frac{1}{2}m|\mathbf{v}_i|^2 = \frac{3}{2}k_BT, \tag{2.4}$$

where T is the temperature at which we want to study the LJ system. We also set initial momentum to zero with the constraint that

$$\frac{1}{N_{atm}}\sum_{i=1}^{N_{atm}}\mathbf{v}_i = 0 \text{ or } \mathbf{v}_{target}. \tag{2.5}$$

Positions and velocities are updated by using equations Eq. (2.2) and Eq. (2.3) and the system is equilibrated at the desired temperature using the Berendsen thermostat [49]. Once the system reaches the equilibrium state, properties are sampled in the micro-canonical ensemble (constant number, volume and energy).

2.2 Force Evaluation

Calculating the force on each particle, due to all other particles in the system, is the most time-consuming part of the algorithm. Accuracy in system properties depends on the accuracy of the force evaluation and the efficiency of the simulation depends on the efficiency of the force evaluation part. We discuss several aspects of the force computation in the following subsections.

2.2.1 Particle-Particle with Finite Cutoff

In the case of a systems of atoms interacting by a pairwise additive potential, all other atoms in the system contribute to the force on atom i. This requires a minimum of $N(N-1)/2$ pair distance computations where N is the number of the atoms. Since typical potentials decay rapidly, they are typically truncated outside some finite cutoff radius r_c, but all pair distances must be computed to determine whether or not any particle is within r_c. This method is very simple to implement but extremely computationally expensive for large number of atoms as it scales as $\mathcal{O}(N^2)$.

2.2.2 Verlet Neighbor List

Maintaining a list of neighbors greatly reduces computational expenses. Force evaluation scaling reduces from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. In this method, a second cutoff radius r_l is introduced which is larger than r_c, and all neighbors separated by less than r_l are stored. We use the same list so long as the displacement of any particle is less than $r_l - r_c$, which makes the calculation $\mathcal{O}(N)$ and assures that all pair particles within r_c are considered in the force computation. As soon as a particle is displaced by more than $r_l - r_c$, we must update the Verlet list which will be of $\mathcal{O}(N^2)$, but needs to be done after every $\approx 10 - 20$ time-steps. This method is simple to implement, but suffers from the error due to finite r_c. For large systems ($\sim 10,000$ atoms), updating Verlet lists using a straight forward $\mathcal{O}(N^2)$ algorithm becomes very time consuming. A cell list algorithm is used when we start to consider large systems. This method forms a Verlet list in $\mathcal{O}(N)$ operations.

2.2.3 Particle-Particle/Particle-Mesh (P^{3}M) Method

The P^{3}M algorithm enables the evaluation of the untruncated potentials with $\mathcal{O}(N \log N)$ operations. The interparticle force is expressed as a sum of two components: $\mathbf{F}_{sr}$ (short range part) and $\mathbf{F}_{lr}$ (long range smooth part). Short range force $\mathbf{F}_{sr}$ has effectively compact support within some cutoff radius r_c and $\mathbf{F}_{lr}$ is a smooth component that can be represented by a rapidly convergent Fourier series. Short range force $\mathbf{F}_{sr}$ is computed by direct particle-particle pair force summation using neighbor lists as discussed above; $\mathbf{F}_{lr}$ is evaluated by solving the Poisson's equation using fast Fourier transforms. This method is discussed in detail in Chapter 5.

2.3 Modeling of the System

2.3.1 Solid-Fluid Interaction

Solid-fluid interactions are modeled with a Lennard-Jones (12,6) potentials with parameters ϵ_{sf} and σ_{sf}. The effect of the solid surface on fluid properties can be best investigated by considering a bulk solid surface. Some researchers simulated solid surface using only 1 - 3 layers of solid atoms due to computational limitations. Such solid surfaces produce a potential which has a faster decaying potential than a realistic solid surface. Israelachvili [50] discussed that interaction potential between a atom and solid surface varies as $-1/D^3$ neglecting the short range term. Considering the computational limitations and need to consider the bulk solid, a equivalent interaction between a solid surface and fluid atom is used.

Consider an FCC(111) (face centered cubic) solid surface like platinum. Fig. 2.1 shows an FCC unit cell. There are two (111) planes between two diagonally opposite atoms, A and B. The unit cell's length is a. The distance between the atoms A and B is $a\sqrt{3}$. We define the lattice constant of an FCC(111) plane as R_0 which is the distance between two adjacent atoms. As face atoms are shared between two unit cells and corner atoms are shared between eight unit cells, we have a total of 4 atoms $(6 \times 1/2 + 8 \times 1/8)$ per unitcell. If dn is the number of solid atoms per unit volume and $dn = 4/a^3$. We can write R_0 as

$$R_0 = \frac{a}{\sqrt{2}}, \tag{2.6}$$

and

$$dn = \frac{4}{2\sqrt{2}R_0^3} = \frac{\sqrt{2}}{R_0^3}. \tag{2.7}$$

We would evaluate the total interaction potential of any atom which is h distance from the solid surface ($-\infty \le x \le \infty, -\infty \le y \le \infty$ and $h \le z \le \infty$). The Lennard-Jones potential between a solid and a fluid atom separated by r is given as

$$\phi_{sf}(r) = 4\epsilon_{sf}\left[\left(\frac{\sigma_{sf}}{r}\right)^{12} - \left(\frac{\sigma_{sf}}{r}\right)^{6}\right], \tag{2.8}$$

where ϵ_{sf} and σ_{sf} are Lennard-Jones energy and length parameters respectively. Let us consider a infinitesimal solid volume $dx\ dy\ dz$ at relative distance (x, y, z) from the fluid atom. The number of atoms in this volume is given by $dn\ dx\ dy\ dz$.

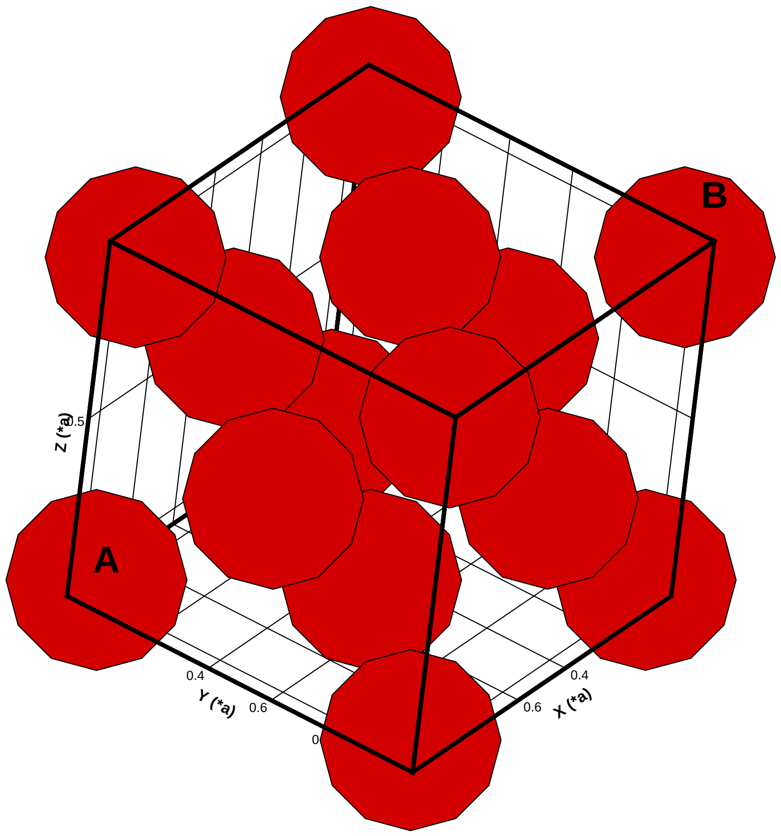


Figure 2.1: FCC unit cell. A and B are two diagonally opposite atoms.

The interaction potential $d\psi$ between a fluid atom and an infinitesimal volume is the number of solid atoms multiplied with the interaction in Eq. (2.8) and it is given by

$$\begin{aligned} d\psi(r) &= dn\; dx\; dy\; dz\; 4\epsilon_{sf}\left[\left(\frac{\sigma_{sf}}{r}\right)^{12} - \left(\frac{\sigma_{sf}}{r}\right)^{6}\right] \qquad (2.9)\\ &= \frac{4\sqrt{2}dx\; dy\; dz}{R_0^3}\epsilon_{sf}\left[\left(\frac{\sigma_{sf}}{r}\right)^{12} - \left(\frac{\sigma_{sf}}{r}\right)^{6}\right], \qquad (2.10)\end{aligned}$$

where

$$r = \sqrt{x^2 + y^2 + z^2}. \qquad (2.11)$$

The total interaction potential between a semi-infinite solid surface and a fluid atom (at distance h from solid surface) is given by

$$\begin{aligned} \psi(h) &= \int_{z=h}^{\infty}\int_{y=-\infty}^{\infty}\int_{x=-\infty}^{\infty} d\psi \qquad (2.12)\\ &= \int_{z=h}^{\infty}\int_{y=-\infty}^{\infty}\int_{x=-\infty}^{\infty} \frac{4\sqrt{2}\epsilon_{sf}}{R_0^3}\left[\left(\frac{\sigma_{sf}}{r}\right)^{12} - \left(\frac{\sigma_{sf}}{r}\right)^{6}\right] dx\; dy\; dz \qquad (2.13)\\ &= \int_{z=h}^{\infty}\int_{y=0}^{\infty}\int_{x=0}^{\infty} \frac{16\sqrt{2}\epsilon_{sf}}{R_0^3}\left[\frac{\sigma_{sf}^{12}}{(x^2+y^2+z^2)^6} - \frac{\sigma_{sf}^{6}}{(x^2+y^2+z^2)^3}\right] dx\; dy\; dz. \qquad (2.14)\end{aligned}$$

First we integrate over x by substituting $x = \sqrt{y^2+z^2}\tan\theta$, and we get

$$\begin{aligned}\psi(h) =& \frac{16\sqrt{2}\epsilon_{sf}}{R_0^3}\int_{z=h}^{\infty}\int_{y=0}^{\infty}\int_{\theta=0}^{\pi/2}\left[\frac{\sigma_{sf}^{12}}{(y^2+z^2)^6\sec^{12}\theta} - \frac{\sigma_{sf}^{6}}{(y^2+z^2)^3\sec^{6}\theta}\right]\times \\ & (y^2+z^2)^{1/2}\sec^2\theta\; d\theta\; dy\; dz \qquad (2.15) \\ =& \frac{16\sqrt{2}\epsilon_{sf}}{R_0^3}\int_{z=h}^{\infty}\int_{y=0}^{\infty}\left[\frac{\sigma_{sf}^{12}}{(y^2+z^2)^{11/2}}\int_{\theta=0}^{\pi/2}\cos^{10}\theta\; d\theta \right. \\ & \left. -\frac{\sigma_{sf}^{6}}{(y^2+z^2)^{5/2}}\int_{\theta=0}^{\pi/2}\cos^{4}\theta\; d\theta\right] dy\; dz. \qquad (2.16)\end{aligned}$$

We use the properties of the beta function $B(m,n)$ defined as

$$\int_0^{\pi/2}\sin^n\theta\cos^m\theta\; d\theta = \frac{1}{2}B\left(\frac{n+1}{2},\frac{m+1}{2}\right), \qquad (2.17)$$

and the beta function is related to gamma function by

$$B(m,n) = \frac{\Gamma(m)\Gamma(n)}{\Gamma(m+n)}. \qquad (2.18)$$

Using the above definition in Eq. (2.16), we get

$$\psi(h) = \frac{16\sqrt{2}\epsilon_{sf}}{R_0^3}\int_{z=h}^{\infty}\int_{y=0}^{\infty}\left[\frac{\sigma_{sf}^{12}B\left(\frac{1}{2},\frac{11}{2}\right)}{2(y^2+z^2)^{11/2}} - \frac{\sigma_{sf}^{6}B\left(\frac{1}{2},\frac{5}{2}\right)}{2(y^2+z^2)^{5/2}}\right] dy\; dz. \qquad (2.19)$$

If we integrate similarly w.r.t y, we get

$$\begin{aligned}
\psi(h) =& \frac{16\sqrt{2}\epsilon_{sf}}{R_0^3} \int_{z=h}^{\infty} \left[\frac{\sigma_{sf}^{12} B\left(\frac{1}{2}, \frac{11}{2}\right) B\left(\frac{1}{2}, 5\right)}{4z^{10}} - \frac{\sigma_{sf}^{6} B\left(\frac{1}{2}, \frac{5}{2}\right) B\left(\frac{1}{2}, 2\right)}{4z^{4}} \right] dz && (2.20) \\
=& \frac{4\sqrt{2}\epsilon_{sf}}{R_0^3} \left[B\left(\frac{1}{2}, \frac{11}{2}\right) B\left(\frac{1}{2}, 5\right) \frac{\sigma_{sf}^{12}}{9h^9} - B\left(\frac{1}{2}, \frac{5}{2}\right) B\left(\frac{1}{2}, 2\right) \frac{\sigma_{sf}^{6}}{3h^3} \right] && (2.21) \\
=& \frac{4\sqrt{2}\epsilon_{sf}}{R_0^3} \left[\frac{\sigma_{sf}^{12}}{9h^9} \frac{\Gamma(\frac{1}{2})\Gamma(\frac{11}{2})}{\Gamma(6)} \frac{\Gamma(\frac{1}{2})\Gamma(5)}{\Gamma(\frac{11}{2})} - \frac{\sigma_{sf}^{6}}{3h^3} \frac{\Gamma(\frac{1}{2})\Gamma(\frac{5}{2})}{\Gamma(3)} \frac{\Gamma(\frac{1}{2})\Gamma(2)}{\Gamma(\frac{5}{2})} \right] && (2.22) \\
=& \frac{4\sqrt{2}\epsilon_{sf}}{R_0^3} \left[\frac{\sigma_{sf}^{12}}{9h^9} \frac{\pi}{5} - \frac{\sigma_{sf}^{6}}{3h^3} \frac{\pi}{2} \right] && (2.23) \\
=& \frac{2\sqrt{2}\pi\epsilon_{sf}\sigma_{sf}^3}{45R_0^3} \left[2\left(\frac{\sigma_{sf}}{h}\right)^9 - 15\left(\frac{\sigma_{sf}}{h}\right)^3 \right]. && (2.24)
\end{aligned}$$

Eq. (2.24) gives the equivalent solid potential due to a semi-infinite FCC solid. Replacing h by z gives,

$$\psi(z) = \frac{2\sqrt{2}\pi\epsilon_{sf}\sigma_{sf}^3}{45R_0^3} \left[2\left(\frac{\sigma_{sf}}{z}\right)^9 - 15\left(\frac{\sigma_{sf}}{z}\right)^3 \right]. \tag{2.25}$$

The force can be obtained by differentiating the potential with distance z as

$$F_s(z) = \frac{2\sqrt{2}\pi\epsilon_{sf}\sigma_{sf}^3}{5R_0^3 z} \left[2\left(\frac{\sigma_{sf}}{z}\right)^9 - 5\left(\frac{\sigma_{sf}}{z}\right)^3 \right]. \tag{2.26}$$

To see the effect of the film thickness on the surface tension, we study a system of LJ atoms placed in a rectangular domain as shown in Fig. 2.2. LJ atoms interact with each other through the LJ potential Eq. (1.12). At the bottom of the domain ($z = 0$), a solid FCC(111) wall is modeled by Eq. (2.25).

The x and y directions are periodic and a mirror boundary condition (atoms reflect spectrally at the boundary) is applied at the top z boundary. Initially

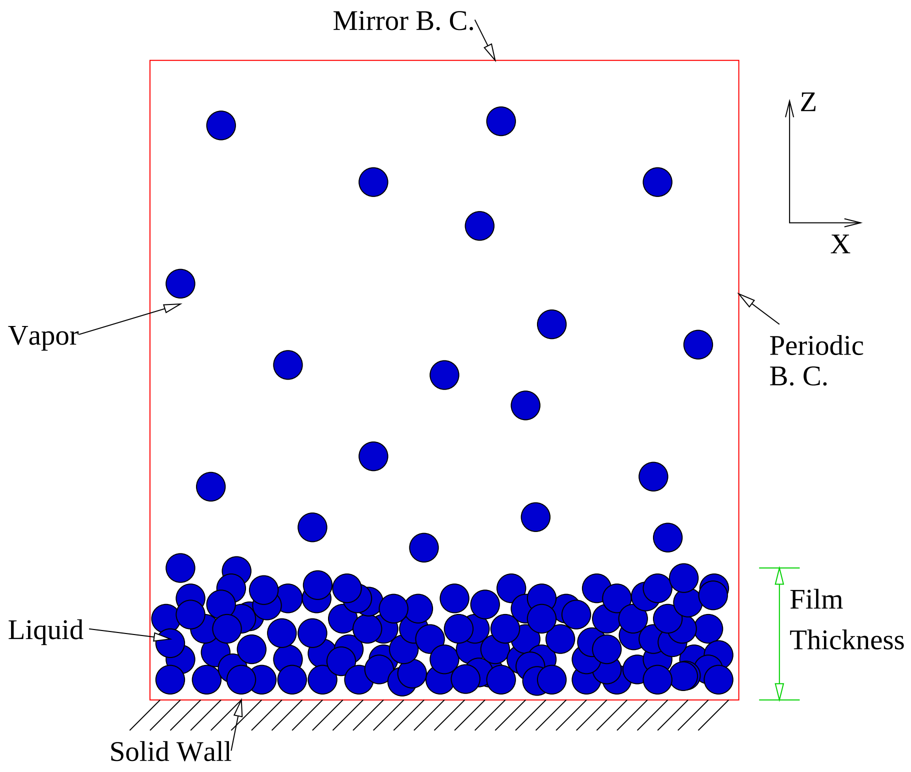


Figure 2.2: Schematic picture of the simulation domain to study the thickness effect on the surface tension.

atoms are placed in the liquid region and the vapor region based on the approximate density corresponding to the temperature using thermodynamical data. This speeds equilibration. Once the equilibrium state is achieved and mean temperature does not change with time, density, pressure and surface tension of the liquid-vapor interface (γ_{lv}) can be sampled.

To study the contact angle dependence on the solid-liquid interaction and temperature, system domain is used as shown in Fig. 2.3. Liquid droplet is simulated in equilibrium with its vapor adjacent to the solid surface. Density is sampled in the space. Density distribution defines the droplet shape and contact angle is measured by drawing the tangent line near the contact line. It is important to adjust the center of mass of the fluid atoms to the center line to achieve good sampling of the density distribution. Simulation details are given in Chapter 4.

2.4 Sampling

Properties are sampled in the MD simulation once the system is equilibrated at a desired temperature. In our simulation, properties are sampled in the *microcanonical ensemble* which is the most convenient for MD. Generally, properties are defined in terms of particles positions, velocities, interatomic potential and forces.

2.4.1 Density

In a MD simulation involving phase equilibrium, a density is required to distinguish liquid-vapor phases. In the case of a liquid-vapor interface parallel to

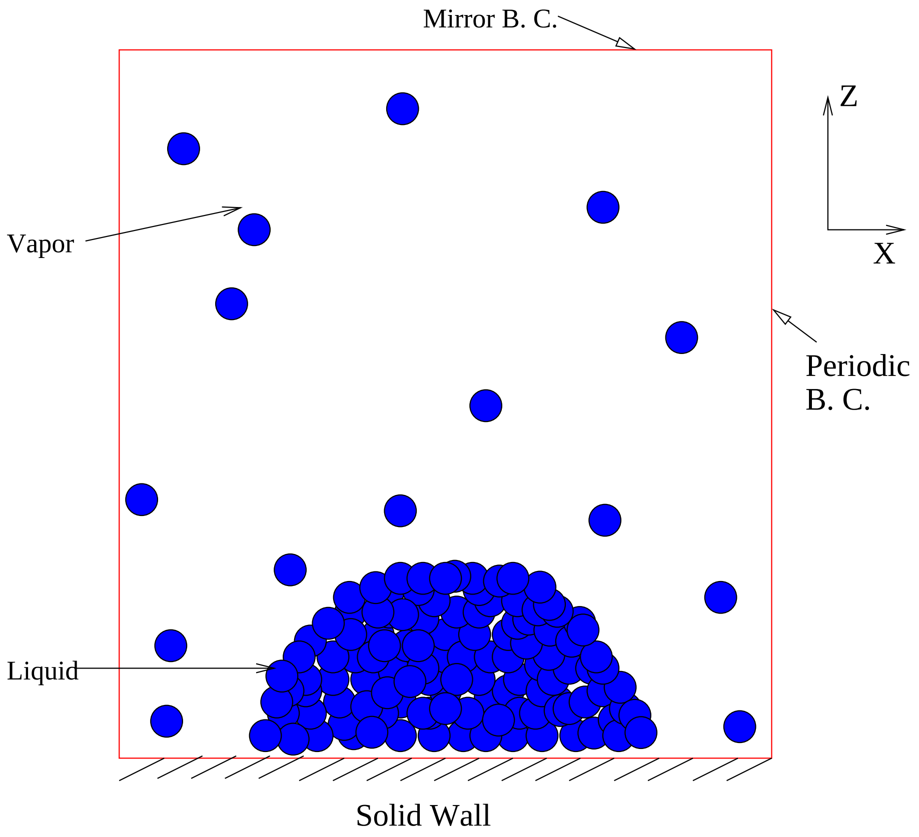


Figure 2.3: Schematic picture of the simulation domain to study contact angle.

$x - y$ plane, we evaluate the density as a function of z by dividing the domain into N_{bin} number of rectangular slabs of thickness Δz. If N_{sample} is the number of times sampled during the simulation and N_{z_i} is the total number of atoms in i^{th} bin accumulated over all samples then the non-dimensional density is given by

$$\rho^*(z_i) = \langle \rho^* \rangle = \frac{N_{z_i} \sigma_{ff}^3}{N_{sample} L_x L_y \Delta z}, \tag{2.27}$$

where L_x and L_y is the box length along the x and y directions. For a three dimensional density distribution as in liquid droplet simulations, bins of size $\Delta x, \Delta y$ and Δz are used and $N_{x,y,z}$ is the total number of atoms in the bin for N_{sample} times. Density is then

$$\rho^*(x, y, z) = \frac{N_{x,y,z} \sigma_{ff}^3}{N_{sample} \Delta x \Delta y \Delta z}. \tag{2.28}$$

2.4.2 Temperature

The temperature of the system is defined as

$$\frac{N_f k_B T}{2} = \frac{E_k}{N_{atm}}, \tag{2.29}$$

where N_f is the number of degrees of freedom per atom, E_k is the total kinetic energy and N_{atm} is the total number of atoms. In the case of LJ atoms, $N_f = 3$ and total kinetic energy is given by

$$E_k = \sum_{i=1}^{N_{atm}} \frac{1}{2} m_i (v_{xi}^2 + v_{yi}^2 + v_{zi}^2). \tag{2.30}$$

2.4.3 Pressure

The pressure tensor is defined in terms of molecular positions and velocities as

$$P_{xy} = \frac{1}{V}\left[\sum_j m_j v_{xj} v_{yj} + \frac{1}{2}\sum_{i,i\neq j}\sum_j r_{xij} f_{yij}\right] \tag{2.31}$$

$$= \frac{1}{V}\left[\sum_j m_j v_{xj} v_{yj} + \sum_{i>j}\sum_j r_{xij} f_{yij}\right], \tag{2.32}$$

where V is the volume element. For a homogeneous single phase system, the pressure components $P_{xx} = P_{yy} = P_{zz}$ are the same. But at a phase boundary, the pressure components are not same, which gives rise to surface tension.

2.4.4 Surface Tension

The Surface tension of an interface is given by Kirkwood and Buff's expression

$$\gamma = \int_{\text{phase1}}^{\text{phase2}} [P_N(z) - P_T(z)]dz. \tag{2.33}$$

For a plane interface perpendicular to z axis, tangential and normal pressure components are defined as

$$P_N(z) = P_{zz}(z), \tag{2.34}$$

and

$$P_T(z) = \frac{P_{xx}(z) + P_{yy}(z)}{2}. \tag{2.35}$$

For pair potentials with $\mathbf{f}_{ij} = f_{ij}\mathbf{r}_{ij}/r_{ij}$, the normal and tangential pressures are

$$\begin{aligned} P_{xx} &= \frac{1}{V}\left[\sum_j m_j v_{xj}^2 + \sum_{i>j}\sum_j r_{xij} f_{xij}\right] \\ &= \frac{1}{V}\left[\sum_j m_j v_{xj}^2 + \sum_{i>j}\sum_j x_{ij}(-\phi_{ij}^{'}\cos\theta_1)\right], \end{aligned} \tag{2.36}$$

$$\begin{aligned} P_{yy} &= \frac{1}{V}\left[\sum_j m_j v_{yj}^2 + \sum_{i>j}\sum_j r_{yij} f_{yij}\right] \\ &= \frac{1}{V}\left[\sum_j m_j v_{yj}^2 + \sum_{i>j}\sum_j y_{ij}(-\phi_{ij}^{'}\cos\theta_2)\right], \end{aligned} \tag{2.37}$$

and

$$P_N(z) = P_{zz} = \frac{1}{V}\left[\sum_j m_j v_{zj}^2 + \sum_{i>j}\sum_j r_{zij} f_{zij}\right], \tag{2.38}$$

where θ_1 and θ_2 are the angles made by $\mathbf{r}_{ij}$ with the x and y axis respectively: $\cos\theta_1 = x_{ij}/r_{ij}$ and $\cos\theta_2 = y_{ij}/r_{ij}$.

Using the above pressure components, we can derive the γ_{lv} to include an external potential ψ due to the solid wall. If we divide the domain into a thin bins of thickness dz parallel to $x-y$ plane, we have $V = A\,dz$, where $A = L_x L_y$ is the area of the plane parallel to $x-y$ plane. Equipartition of kinetic energy gives

$$\left\langle \frac{1}{2}\rho(z)A\,dz \sum_j v_{xj}^2 \right\rangle = \left\langle \frac{1}{2}\rho(z)A\,dz \sum_j v_{yj}^2 \right\rangle = \left\langle \frac{1}{2}\rho(z)A\,dz \sum_j v_{zj}^2 \right\rangle \tag{2.39}$$

$$= \frac{1}{2}kT\langle\rho(z)\rangle A\,dz. \tag{2.40}$$

Also,

$$\left\langle \frac{1}{V} \sum_j m_j v_{xj}^2 \right\rangle = \left\langle \frac{1}{A\,dz} \rho(z)\, A\, dz \sum_j v_{xj}^2 \right\rangle = kT \langle \rho(z) \rangle \text{ (from Eq. 2.40).} \tag{2.41}$$

So we can write Eq. (2.36) and Eq. (2.37) as

$$P_{xx}(z) = kT\langle \rho(z) \rangle - \frac{1}{A\,dz} \sum_{i>j} \sum_j \frac{x_{ij}^2}{r_{ij}} \phi_{ij}^{'}, \tag{2.42}$$

$$P_{yy}(z) = kT\langle \rho(z) \rangle - \frac{1}{A\,dz} \sum_{i>j} \sum_j \frac{y_{ij}^2}{r_{ij}} \phi_{ij}^{'}, \tag{2.43}$$

and

$$P_T(z) = \frac{P_{xx}(z) + P_{yy}(z)}{2} = kT\langle \rho(z) \rangle - \sum_{i>j} \sum_j \frac{x_{ij}^2 + y_{ij}^2}{2A\,dz\, r_{ij}} \phi_{ij}^{'}, \tag{2.44}$$

where $\phi^{'}$ is $\frac{d\phi}{dr_{ij}}$. Now Eq. (2.38) becomes

$$P_N(z) = kT\langle \rho(z) \rangle - \sum_{i>j} \sum_j \frac{z_{ij} \phi_{ij}^{'} \cos\theta_3}{A\,dz} - \sum_j \frac{z_j \psi^{'}(z)}{A\,dz} \tag{2.45}$$

The last term in Eq. (2.45) is the influence of the wall: $\psi^{'} = \frac{d\psi}{dz}$. The angle between $\mathbf{r_{ij}}$ and the z axis is θ_3 which is given as

$$\cos\theta_3 = \frac{z_{ij}}{r_{ij}}. \tag{2.46}$$

From Eq. (2.45) and Eq. (2.46), we obtain

$$P_N(z) = kT\langle \rho(z) \rangle - \sum_{i>j} \sum_j \frac{z_{ij}^2 \phi_{ij}^{'}}{A\, r_{ij}\, dz} - \sum_j \frac{z_j \psi^{'}(z)}{A\,dz}, \tag{2.47}$$

thereafter Eq. (2.44) and Eq. (2.47) yield

$$P_N(z) - P_T(z) = \sum_{i>j} \sum_{j} \frac{x_{ij}^2 + y_{ij}^2 - 2z_{ij}^2}{2A\, r_{ij}\, dz} \phi_{ij}^{'} - \sum_{j} \frac{z_j \psi^{'}(z_i)}{A\, dz}. \tag{2.48}$$

Thus the liquid-vapor interfacial tension Eq. (2.33) is

$$\gamma_{lv} = \left\langle \sum_{i>j} \sum_{j} \frac{x_{ij}^2 + y_{ij}^2 - 2z_{ij}^2}{2A\, r_{ij}} \phi_{ij}^{'} - \sum_{j} \frac{z_j \psi^{'}(z)}{A}, \right\rangle \tag{2.49}$$

which is similar to the solid-fluid surface tension as derived by Navascués and Barry [40] using statistical mechanics and mechanical balance of liquid adjacent to the wall. In absence of any external field, the above expression reduces to

$$\gamma_{lv} = \left\langle \sum_{i>j} \sum_{j} \frac{x_{ij}^2 + y_{ij}^2 - 2z_{ij}^2}{2A\, r_{ij}} \phi_{ij}^{'} \right\rangle \tag{2.50}$$

which has been used in the literature for a liquid slab in equilibrium with its vapor on both sides.

Many of these statistical quantities are sensitive to the cutoff radius r_c. Chapela *et al.* [17] derived the tail correction due to finite r_c by assuming the density profile near the transition as hyperbolic tangent profile given by

$$\rho(z) = \frac{\rho_l + \rho_v}{2} - \frac{\rho_l - \rho_v}{2} \tanh\left(\frac{2(z - z_0)}{d}\right), \tag{2.51}$$

where

$$d = -\left.\frac{(\rho_l - \rho_v)}{d\rho/dz}\right|_{z=z_0}, \tag{2.52}$$

is a measure of interfacial thickness. Chapela's expression for the tail correction is

$$\gamma_{tail} = \int_0^1 \int_{r_c}^{\infty} 12\pi \left(\rho_l - \rho_v\right)^2 \coth\left(\frac{2rs}{d}\right)\left(\frac{3s^3 - s}{r^3}\right) dr\, ds \tag{2.53}$$

as corrected by Blokhius *et al.* [18].

The tail correction is added to the surface tension value Eq. (2.50) from the simulation. Density is sampled and ρ_l, ρ_v and d are obtained from the density profile. Eq. (2.53) are evaluated numerically. Density profile is very sensitive to r_c and it is not corrected. Thus γ_{tail} has an error too. As we increase r_c, properties and γ_{tail} should have much less error which is shown in the next chapter.

2.5 Hamaker Constant

Van der Waals forces are the long range forces between two bodies that arise due to the mutual orientation of two molecules. Hamaker [51] followed a summation procedure which assumes that the van der Waals energy is pairwise additive and he developed the total interaction energy between two large bodies in the free space. London [52] explained the origin of the inverse sixth power dependence of interatomic energy. These forces govern the physical phenomena and can be characterized by the Hamaker constant (A) (Israelachvili [50] which is defined as

$$A = \pi^2 C \rho_1 \rho_2. \tag{2.54}$$

The Hamaker constant is defined for two bodies of density ρ_1 and ρ_2 and C is the constant in dispersion component of the interbody potential. For the atom-solid surface interaction as shown in Fig. 2.4, the interaction energy is

$$\psi = -\frac{\pi C \rho_1}{6z^3}, \tag{2.55}$$

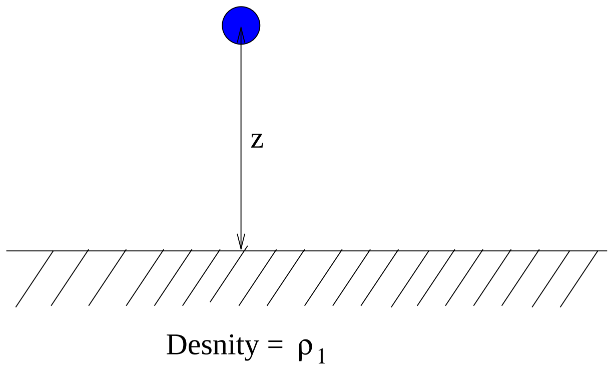


Figure 2.4: Interaction between a atom and solid surface.

where z is the distance between the atom and the solid surface. From Eq. (2.54) and Eq. (2.55), ψ can be expressed in terms of Hamaker constant as

$$\psi = -\frac{A}{6\pi\rho_2 z^3}. \tag{2.56}$$

This term is the same as the dispersive long range part in Eq. (2.25). Hamaker constant can be found by equating these two terms as

$$\psi = -\frac{A}{6\pi\rho_2 z^3} = \frac{2\sqrt{2}\pi}{3}\frac{\epsilon_{sf}\sigma_{sf}^6}{R_0^3 z^3}. \tag{2.57}$$

Solving for A gives

$$A = 4\sqrt{2}\pi^2\frac{\epsilon_{sf}\sigma_{sf}^6\rho_2}{R_0^3}. \tag{2.58}$$

Since ρ_2 is the number density of the fluid (number of atoms per unit volume), the Hamaker constant has unit of energy. Its typical value is about 10^{-19} J for most liquid-solid interactions.

CHAPTER 3

SURFACE TENSION OF LENNARD JONES FLUID

3.1 Liquid-Vapor Equilibrium without External Field

A free liquid-vapor interface of a LJ fluid was simulated to validate the MD algorithm. The simulation domain was a triply periodic rectangular box. The initial condition was a liquid slab in the middle of the simulation box. The system was equilibrated at the desired temperature for 50,000 time steps by velocity rescaling and the 25,000 time steps of non-thermostat equilibration. Properties were sampled (density and surface tension) for an additional 425,000 time-steps. Simulation parameters in non-dimensional from are given in Table 3.1 with dimensional parameters that correspond to Argon (ϵ/k_B = 119.8 K, σ = 3.405x10^{-10} m, $\mathcal{M}$ = 0.03994 kg/mol).

Simulations were performed with cutoff radii of 2.5σ, 4.5σ and 6.5σ. The effect of cutoff radius is shown in Fig. 3.1 at temperature of 0.70 and 0.85 respec-

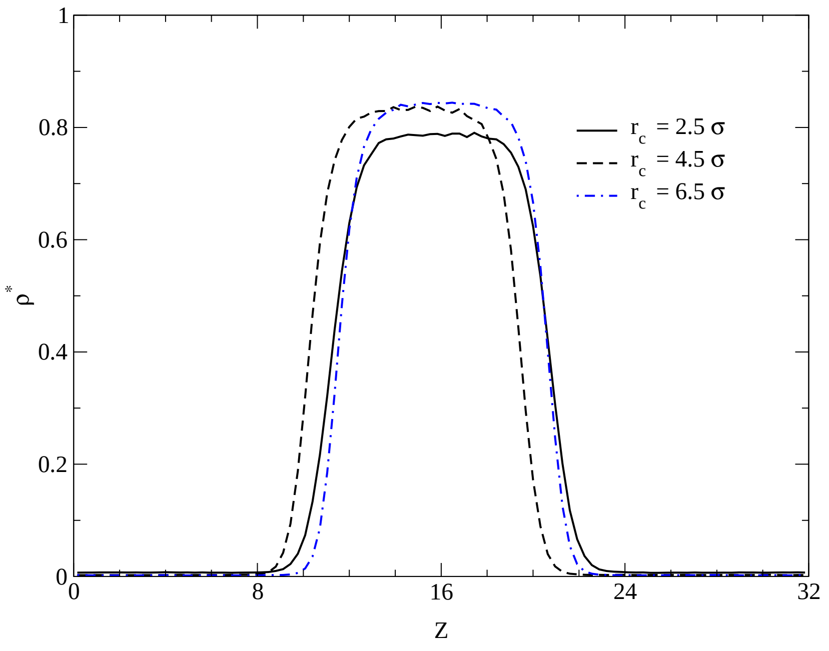
Table 3.1: Simulation parameters

Parameters	Non-Dimensional	Argon
Temperature	$T^* = 1$	$T = 119.8$ K
Density	$\rho^* = 0.7$	$\rho = 1176$ kg/m^3
Time-step	$\Delta t^* = 0.005$	$\Delta t = 1.09$x10^{-14} s

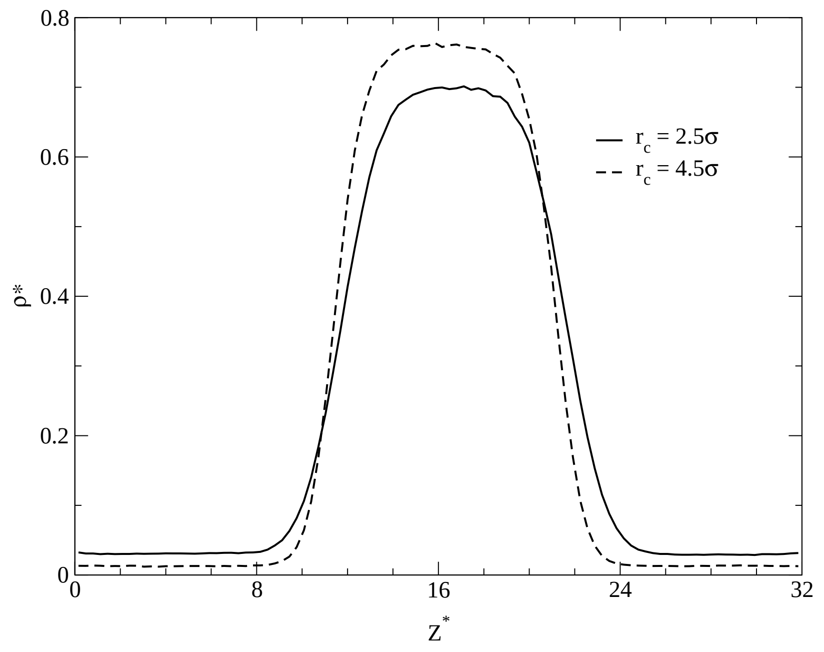
tively. Clearly, the density profile is very sensitive to cutoff radius. Higher liquid density and lower vapor density is observed at larger r_c. Larger r_c is expected to give higher accuracy but computational operations scale as Nr_c^3, making large r_c expensive for large systems. Densities are compared with experimental data and Trokhymchuk and Alejandre [22] results in Fig. 3.2. Their results for $r_c = 2.5\sigma$ match with our results at the same cutoff radius. Also, we observe that densities are closer to the experimental data of Ar as r_c is increased. However, we see liquid and vapor densities are higher and lower respectively at larger cutoff radius. This can be explained by the change in the density distribution near the interface. At higher cutoff radius, more vapor atoms near the interface is pulled towards the liquid atoms because long range dispersive part of LJ force is extended to a higher cutoff radius. This causes lower vapor density and higher liquid density. Densities are found sensitive to r_c.

The surface tension of liquid-vapor interface is calculated by Eq. (2.50). It should be noted that sampling is done over all atom pairs i and j such that j atom is the closest periodic image to i atom. Since we have two liquid-vapor interfaces in this geometry, surface tension of the interface is half of the total summation value in Eq. (2.50).

Simulation results are compared with thermodynamical correlations based on the principle of corresponding states for the surface tension of Argon in Fig. 3.3. Open symbols are simulation results without tail corrections and filled symbols have tail corrections Eq. (2.53). Here we see that a lower surface tension than the thermodynamical correlation is predicted at $r_c = 2.5\sigma$. The tail correction cannot fully compensate for r_c because the density profile and the interfacial thickness are sensitive to r_c too. From Fig. 3.2 we observe lower $(\rho_l - \rho_v)$ and higher d values for smaller r_c. However, for $r_c = 4.5\sigma$ and 6.5σ, tail correction

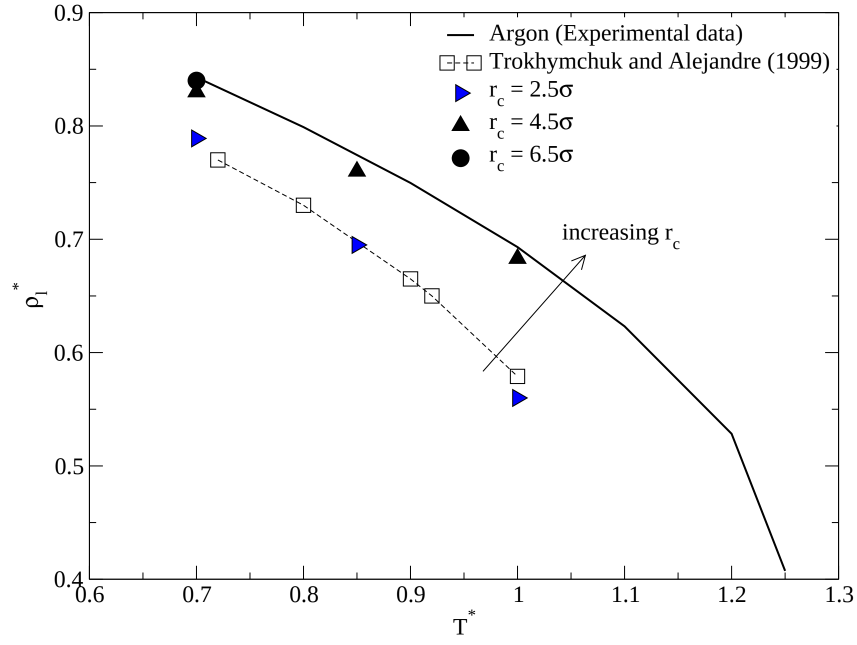


(a)

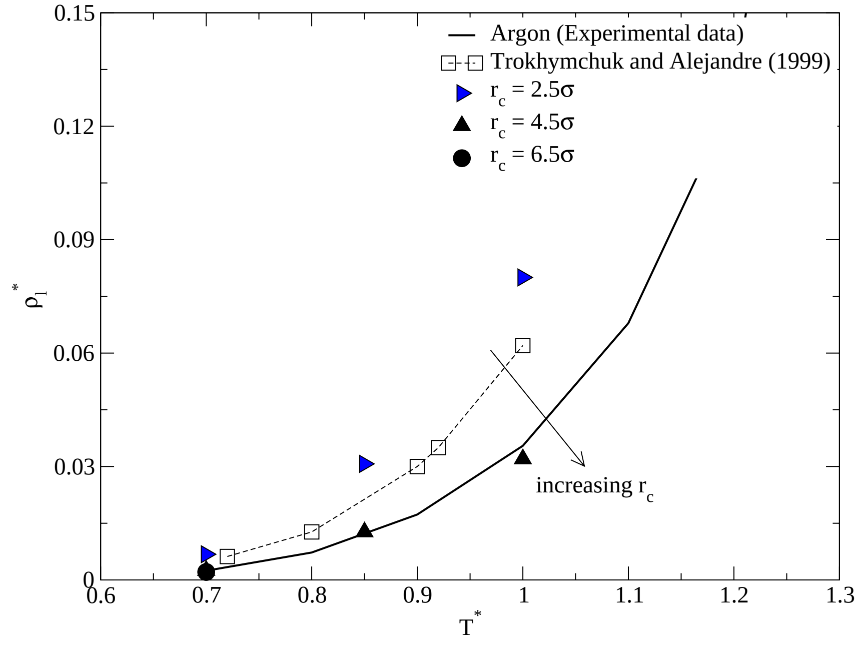


(b)

Figure 3.1: Effect of cutoff radius on the density profile at various temperatures, (a) $T^{*} = 0.70$ and (b) $T^{*} = 0.85$



(a)



(b)

Figure 3.2: Comparison of (a) the liquid density and (b) the vapor density with literature data as a function of temperature.

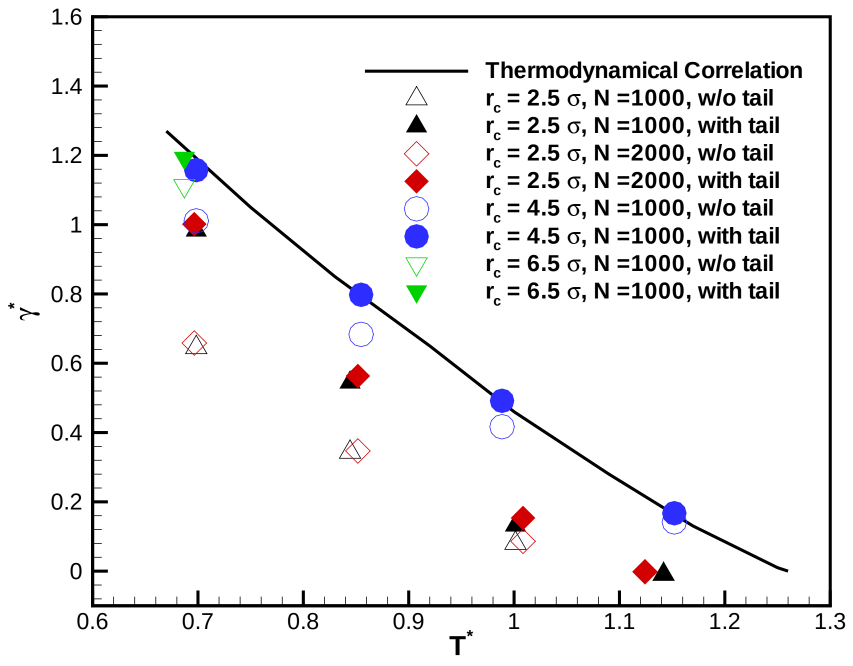


Figure 3.3: The surface tension of the liquid-vapor interface in a liquid slab geometry.

is successful. These computer experiments show that we need at least $r_c = 4.5\sigma$ with appropriate tail correction to estimate the surface tension for $r_c \rightarrow \infty$.

3.2 Liquid-Vapor Equilibrium Adjacent to A Solid Wall

To study the wall effect on the surface tension, we included a solid wall at $z = 0$. A bulk potential as in Eq. (2.25) is used. Initially, a liquid layer is placed near $z = 0$ and, liquid and vapor atoms are distributed based on the approximate density of each phase at the desired temperature. The system is equilibrated using Berendsen's thermostat for 100,000 time-steps:

$$\mathbf{v}_i^{new} = C\mathbf{v}_i^{old}, \tag{3.1}$$

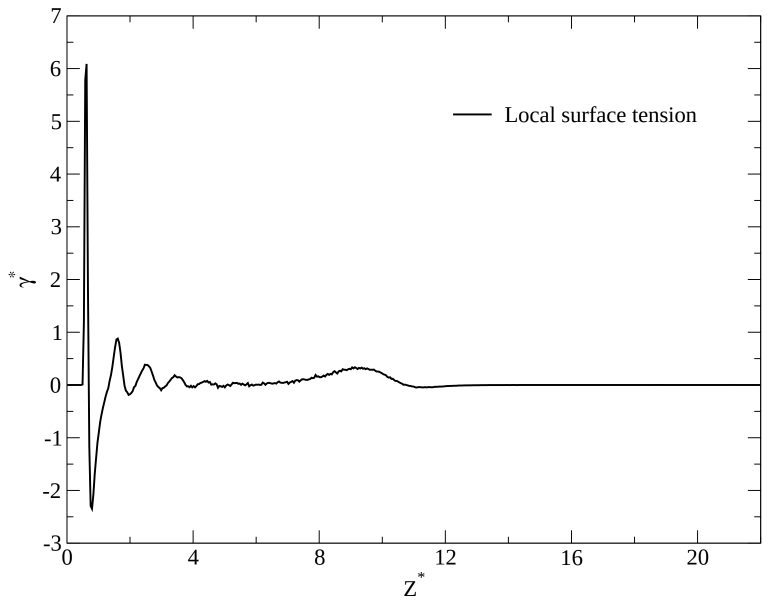
with

$$C = \sqrt{1 + \frac{\Delta t^*}{\tau}\left(\frac{T_{old}}{T_{target}} - 1\right)}, \tag{3.2}$$

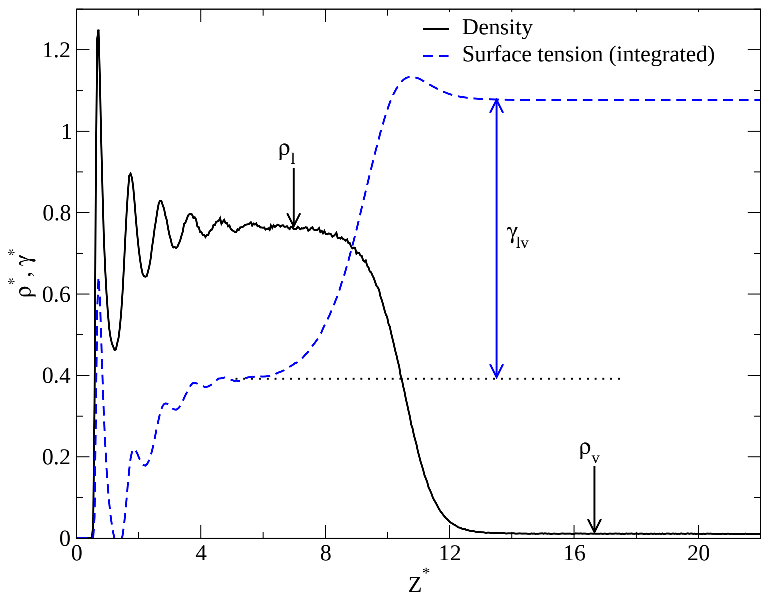
where $\tau = 20\Delta t^*$ and $\Delta t^* = 0.005$. This is followed by 50,000 time steps of non-thermostat equilibration before sampling. Sampling is performed every 50 time steps for the next 1 million time-steps. As in the case of the free liquid slab, we found that $r_c = 4.5\sigma$ with the tail correction gives correct surface tension values, so $r_c = 4.5\sigma$ is taken for this case. A neighbor list is made with $r_l = 5.0\sigma$. Wall parameters used in Eq. (2.25) are $\sigma_{sf} = 0.8\sigma_{ff}$, $\epsilon_{sf} = 0.490\epsilon_{ff}$ and $R_0 = 0.8147\sigma_{ff}$.

Properties (density and surface tension) are sampled by dividing the domain into 512 xy slabs. The local surface tension is evaluated by calculating the difference between tangential and normal components in each slab. This is non-zero only near the interface as shown in Fig. 3.4(a). The local surface tension can be integrated with distance z to obtain the integrated surface tension profile. The surface tension of the liquid-vapor interface is the increase in the integrated surface tension value across the interface as shown in Fig. 3.4(b).

In Fig. 3.4 an external potential due to the solid wall is applied at $Z^* = 0$. We see that fluctuations of the liquid density decay with Z^* near the solid wall and approach a constant liquid density. As Z^* increases further, there is a smooth transition from the liquid to the vapor region. Pressure components (normal and tangential) differ from each other due to non symmetric density profile near the transition region and this effect gives rise to the surface tension. Since a cutoff



(a)



(b)

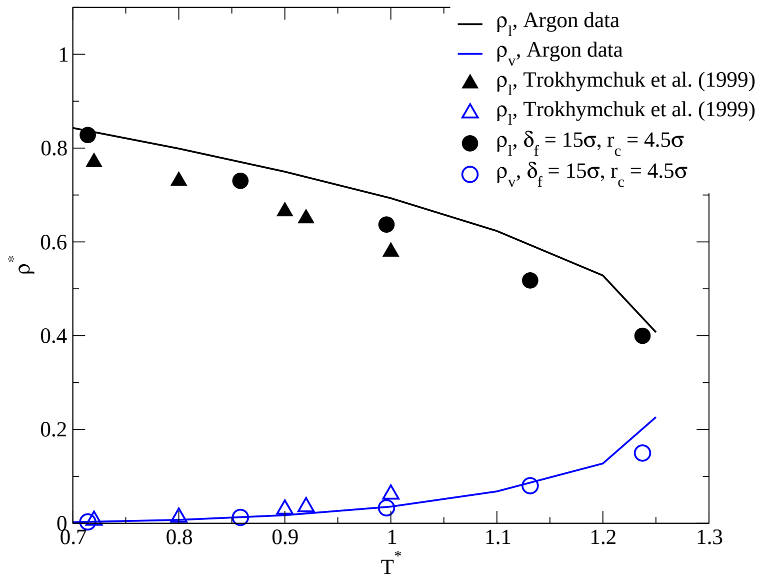
Figure 3.4: At $T^* = 0.85$, (a) local surface tension profile (b) density profile and the integrated surface tension profile.

radius ($= 4.5\sigma$) is used in the simulation, tail correction of the surface tension needs to be added to predict the correct surface tension value. Densities, ρ_l, ρ_v, and the interfacial thickness, d are obtained from the simulation result and used in the tail correction expression.

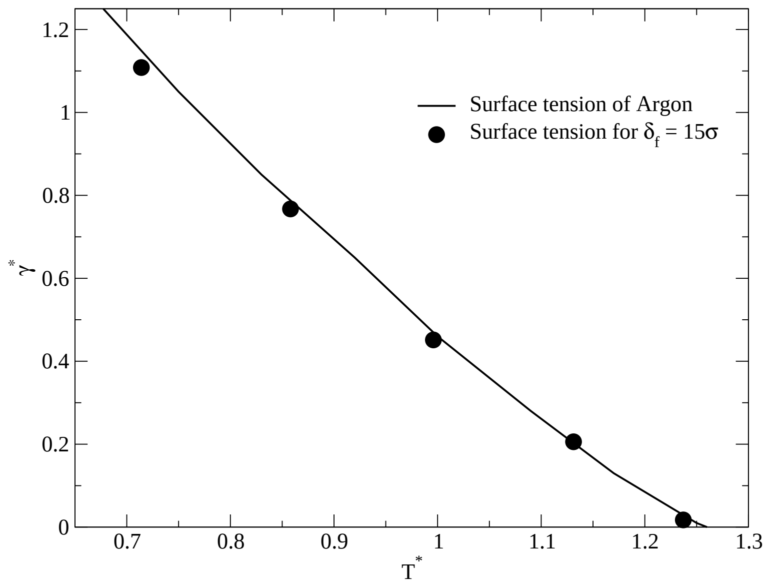
The film thickness (δ_f) was varied from 5σ to 15σ by changing the cross section area of the domain and keeping number of atoms fixed. The system was simulated at five temperatures ranging from the triple point ($T^* = 0.7$) to the critical temperature ($T^* \approx 1.26$).

First we compare the liquid and the vapor density with experimental data and data in the literature at various temperatures for the thick $\delta_f (= 15\sigma)$ film. Fig. 3.5 compares our estimates for the coexisting densities to those given by Trokhymchuk and Alejandre's [22] and experimental data. The surface tension data (Fig. 3.5(b)) is compared with the surface tension of Argon fluid. It shows that presence of the solid wall has no effect for $\delta_f \approx 15\sigma$.

To see the wall effect, liquid films of various δ_f is simulated. At lower δ_f ($\leq 5\sigma$) , the film breaks up and the surface is partially wet. At higher film thickness, an uniform film is formed. It can be observed in Fig. 3.6 which is the $x - y$ view of the domain. The two cases have different surface area keeping the same number of atoms. Fig. 3.6(a) has surface area $14 \times 14\sigma^2$ and Fig. 3.6(b) has $18.08 \times 18.08\sigma^2$ area. The first case produces an uniform film of thickness 7.32σ whereas the second case corresponds to a non-uniform film. Increasing the surface area while keeping the same number of atoms forces a thinner film in 2D periodic domain, but a thinner film is not sustainable and breaks up. The solid surface has a dryout region in some region. Due to non-uniformities of the liquid film, the surface tension and the liquid density can not be measured for this case. For $\epsilon_r = 0.490\epsilon$ and 0.653ϵ and surface area of $18.08 \times 18.08\sigma^2$ or larger, the liquid



(a)



(b)

Figure 3.5: For thick film ($\delta_f \approx 15\sigma$) (a) comparison of density (b) comparison of the surface tension.

film is found to be unstable and breaks up at temperature = 0.70, 0.85 and 1.00. In these cases, the films are non-uniform as in Fig. 3.6(b) or form cylindrical droplets aligned with x or y co-ordinate.

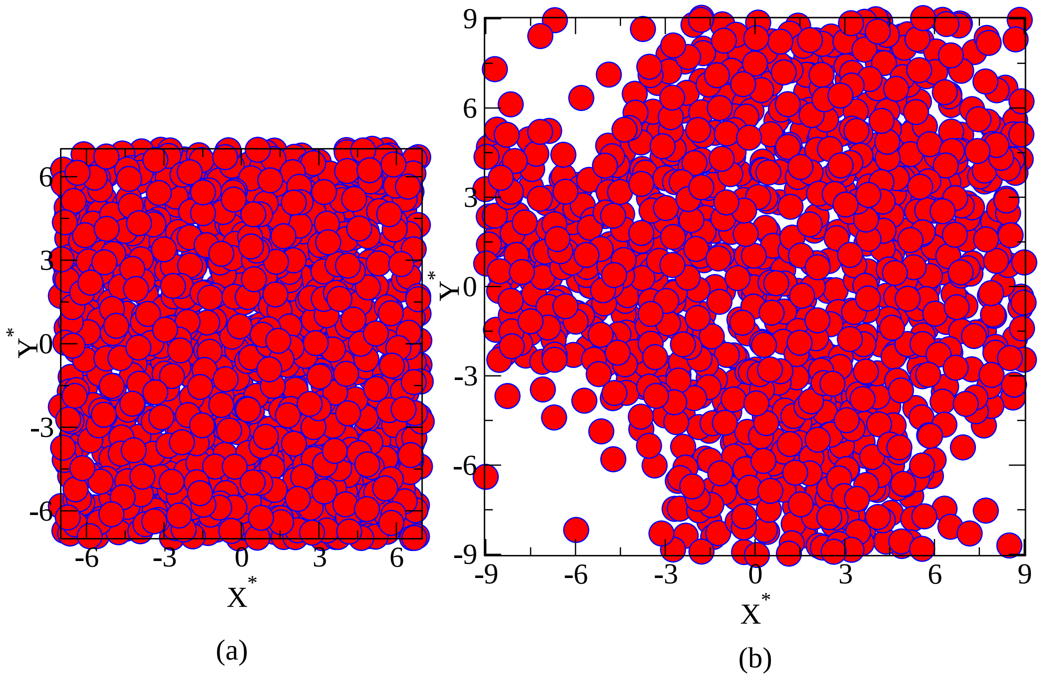


Figure 3.6: Top view of the liquid film, domain cross sectional surface area is (a) $14 \times 14\sigma^2$ (b) $18.08 \times 18.08\sigma^2$. $T^* = 0.85$

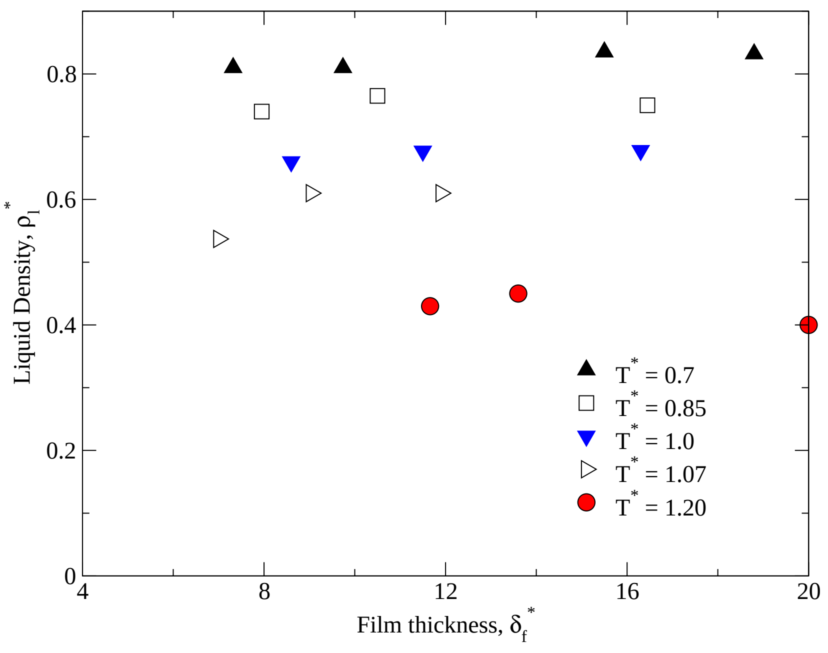
Fig. 3.7 shows the variation of the liquid and the vapor densities with the film thickness at several temperatures. This shows that the liquid and the vapor densities change very little (less than 5%) with temperature. These densities correspond to the macroscopic densities of the fluid. As the film is not stable for thinner ($\sim 5\sigma$) liquid films, liquid density is not evaluated. The vapor density is found to be independent of the film thickness. Fig. 3.8 shows the variation of the surface tension with the film thickness. The surface tension is found to be weakly dependent on the film thickness. From this observation, we concluded that densities and interfacial tensions do not depend on film thickness for $\delta_f > 6\sigma$.

The liquid film is unstable for film thickness $< 6\sigma$ and $\epsilon_{sf} = 0.49\epsilon$ and 0.65ϵ, but can sustain a uniform film thickness at higher solid-fluid strength ϵ_{sf} and $\delta_f = 4.0\sigma \sim 5.0\sigma$. Thin liquid film is simulated at higher $\epsilon_{sf} = 0.816\epsilon$ and 0.98ϵ. For surface of $18.08 \times 18.08\sigma^2$, uniform film is observed with $\delta_f = 4.0\sigma \sim 5.0\sigma$. However, the film breaks up for the surface area of $22.6 \times 22.6\sigma^2$, suggesting that an uniform thinner film is not possible for $\epsilon_{sf} = 0.816\epsilon$ and 0.98ϵ. Table 3.2 gives the density and surface tension of uniform films with the cross sectional area of $18.08 \times 18.08\sigma^2$.

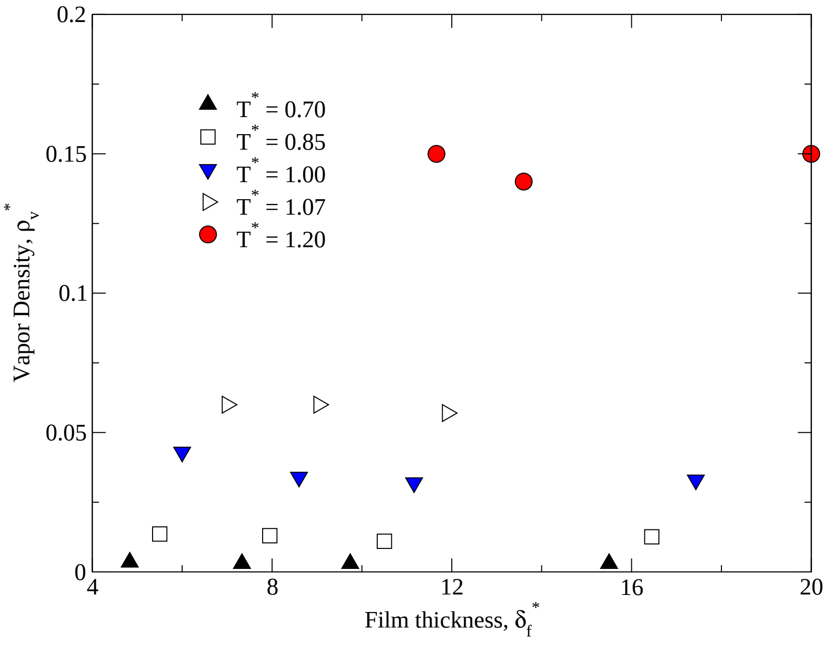
Table 3.2: Densities and the surface tension of thin films, $\sigma_{sf} = 0.80\sigma$

T^*	ϵ_{sf}/ϵ	Thickness (δ_f^*)	ρ_l^*	ρ_v^*	γ^*
0.7	0.816	4.0	0.78	0.0031	1.154
0.85	0.816	4.12	0.76	0.0115	0.766
1.00	0.816	4.28	0.58	0.0416	0.3041
0.7	0.980	4.12	0.80	0.0028	1.0803
0.85	0.980	4.24	0.76	0.0136	0.7648
1.00	0.980	4.67	0.60	0.0144	0.2937

These liquid films are also investigated for $\sigma_{sf} = 0.863436\sigma$ which corresponds to the Argon-Platinum interaction. Increasing σ_{sf} increases the solid-fluid attraction force term. Energy parameters $\epsilon_{sf} = 0.490\epsilon, 0.653\epsilon$ and 0.816ϵ are used. The temperature is varied from 0.70 to 1.07 and the film thickness δ_f is varied from 4σ to 12σ by changing the cross sectional area and keeping the same number of atoms. An uniform film of thickness $4\sigma \sim 5\sigma$ is not sustainable at the lowest ϵ_{sf} ($= 0.490\epsilon$) and breaks up. However an uniform film of thickness $4\sigma \sim 5\sigma$ is sustained at higher ϵ_{sf} (0.653ϵ and 0.816ϵ). For these cases, densities and surface tension values are tabulated in Table 3.3. For such a thin uniform film, the



(a)



(b)

Figure 3.7: Variation of (a) the liquid density and (b) the vapor density with the film thickness, δ_f. $\epsilon_{sf} = 0.49\epsilon$, $\sigma_{sf} = 0.8\sigma$.

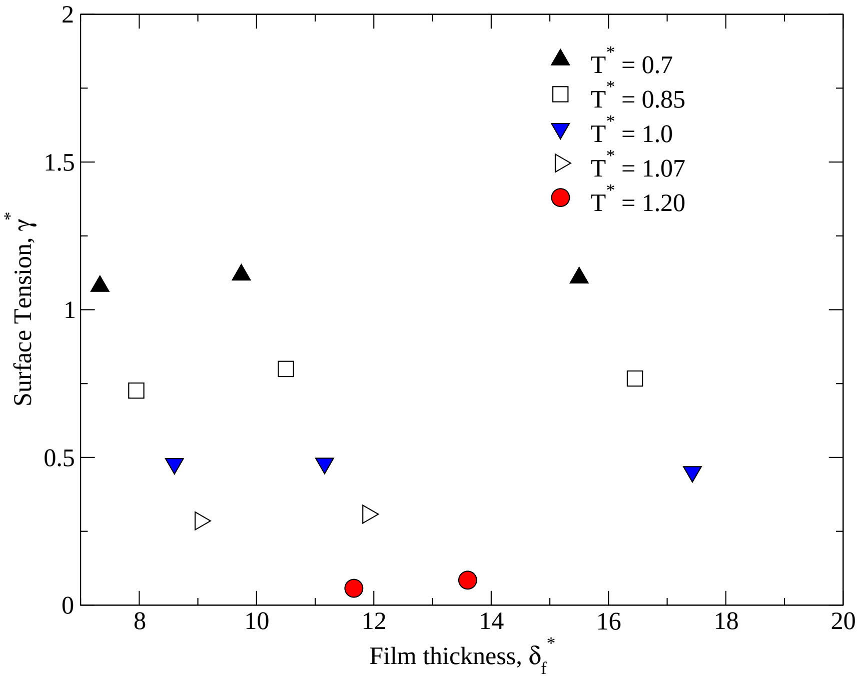


Figure 3.8: Variation of the surface tension with film thickness, δ_f. $\epsilon_{sf} = 0.49\epsilon$, $\sigma_{sf} = 0.8\sigma$.

surface tension of the interface is evaluated by integrating the difference between the normal and tangential pressure components across the interface, but a bulk liquid region is not well defined due to the fluctuations in the density profile and has large error associated with the surface tension. However, we observed a much lower surface tension (30 % lower than the macroscopic value).

Table 3.3: Densities and surface tension of thin films, $\sigma_{sf} = 0.863\sigma$

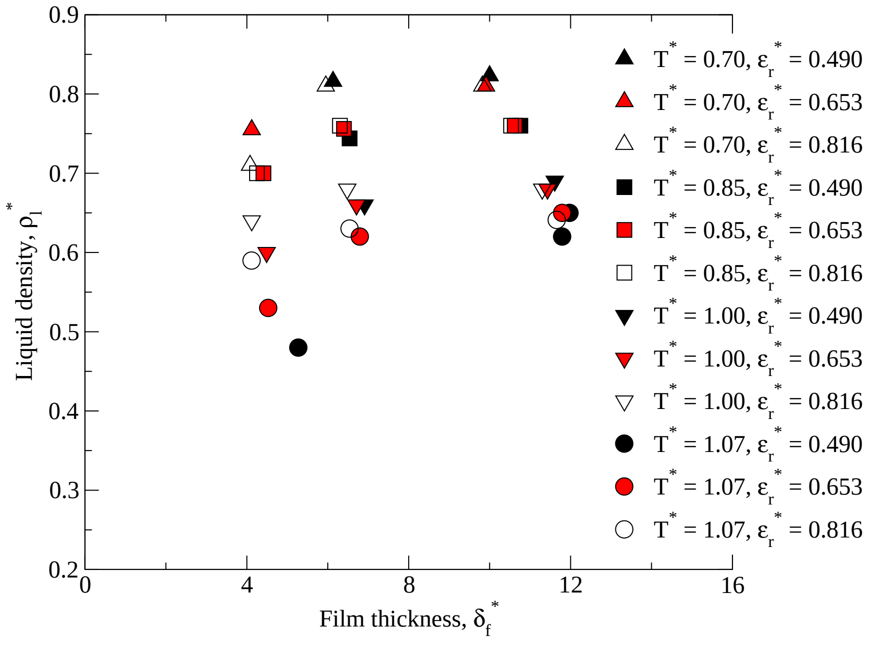
T^*	ϵ_{sf}/ϵ	Thickness (δ_f^*)	ρ_l^*	ρ_v^*	γ^*
0.70	0.653	4.12	0.755	0.0026	0.7962
0.85	0.653	4.41	0.7	0.0115	0.5215
1.00	0.653	4.49	0.6	0.0371	0.3143
1.07	0.653	4.53	0.53	0.0493	0.2008
0.70	0.816	4.08	0.71	0.0025	0.7693
0.85	0.816	4.25	0.7	0.0111	0.4963
1.00	0.816	4.12	0.64	0.0367	0.3284
1.07	0.816	4.12	0.59	0.049	0.2055

A liquid film of varying thickness is studied for $\sigma_{sf} = 0.863436\sigma$. Fig. 3.9 shows the variation of the liquid and the vapor densities with the film thickness for different values of the parameter ϵ_{sf}. The density of the vapor is independent of the film thickness and ϵ_{sf}. Even though all these cases have a uniform film adjacent to the solid surface, we notice that the liquid density decreases for film thickness $\leq 5\sigma$. As the film gets thinner, density fluctuations do not decays completely and the liquid density does not reach the bulk density value. This gives a lower density value than the macroscopic value. Fig. 3.10 shows the density of the first layer of the liquid adjacent to the solid surface and its variation with film thickness and ϵ_{sf}. Peak density of the monolayer shows a weak dependence on the film thickness, but varies significantly with the ϵ_{sf}. Higher energy surface produces higher peak density of the monolayer adjacent to the surface. At temperature $T^* = 0.70$, peak density increases from 2.11 to 3.50 by increasing the

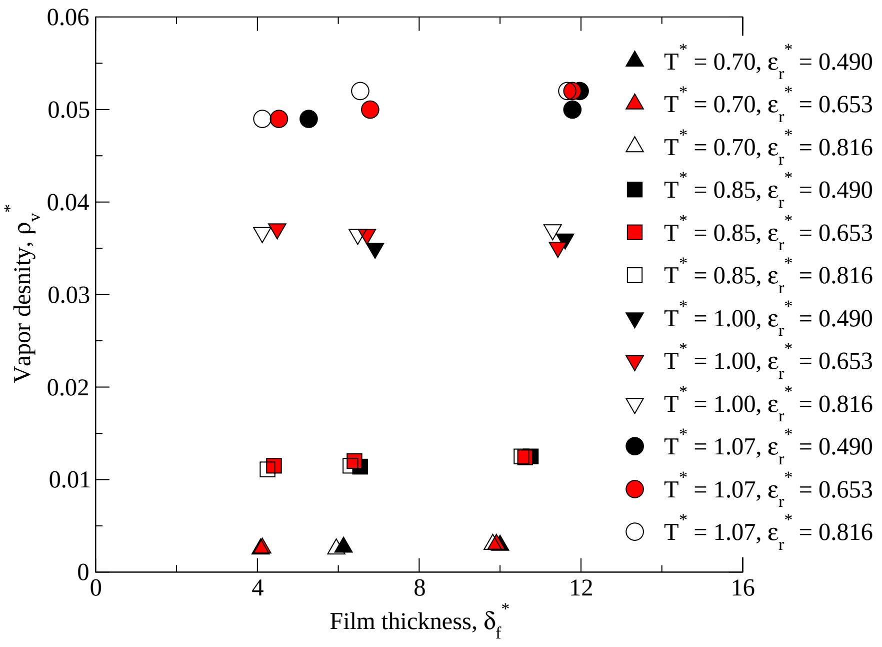
ϵ_{sf} from 0.490 to 0.816. It is distinctly observed at other temperatures too.

The surface tension of the liquid-vapor interface is evaluated and plotted as a function of film thickness in Fig. 3.11. The surface tension is constant for film thickness $\geq 7\sigma$ but decreases by as much as 20 % when film thickness is less than 5σ. The decrease in surface tension is due to the lower density of the bulk liquid for film thickness $\leq 5\sigma$. As the transition from a lower liquid density to a macroscopic vapor density produces a smaller difference between tangential and normal pressure components, we get a lower value of the surface tension.

In this chapter, we investigated the thin film properties adjacent to the solid surface. Liquid and vapor densities are compared with the macroscopic value of the argon fluid and literature data. Agreement is satisfactory. Interfacial tension does not depend on the film thickness for $\delta_f \geq 7\sigma$, but decreases by 20 % for thinner films. Higher values of ϵ_{sf} and σ_{sf} have strong effects on the stability of thin films with $\delta_f \leq 5\sigma$. This gives us important information in choosing the right solid and liquid combination to achieve coating of the liquid film of thickness 1.5 nm. In all cases, we found that the peak density of the monolayer of liquid adjacent to the solid surface increases with increasing ϵ_{sf}.



(a)



(b)

Figure 3.9: Variation of (a) the liquid density and (b) the vapor density with the film thickness, δ_f. $\sigma_{sf} = 0.863436\sigma$.

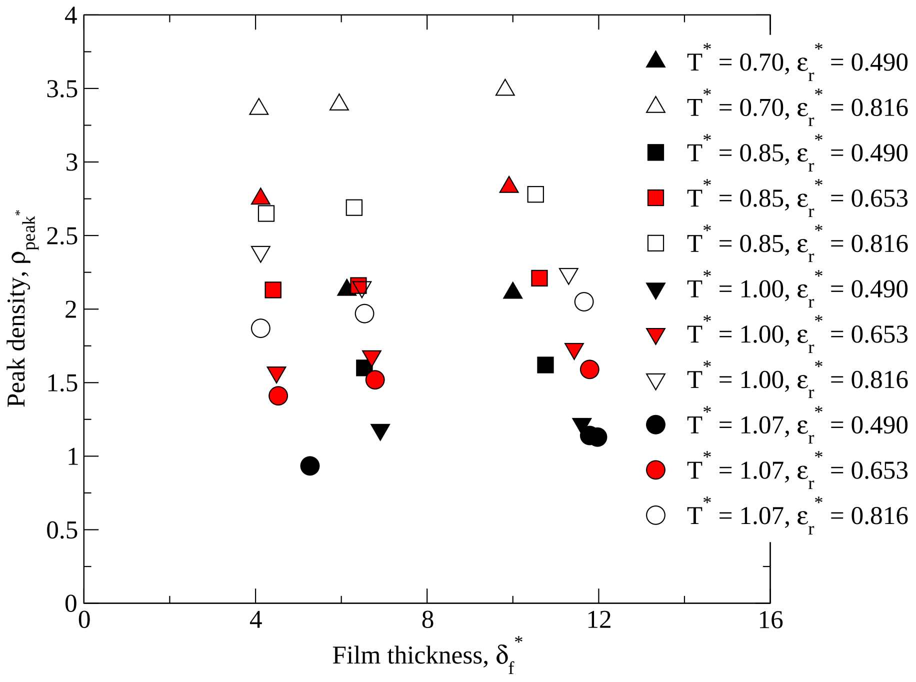


Figure 3.10: Variation of the density of the first liquid layer adjacent to the solid surface with δ_f. $\sigma_{sf} = 0.863436\sigma$.

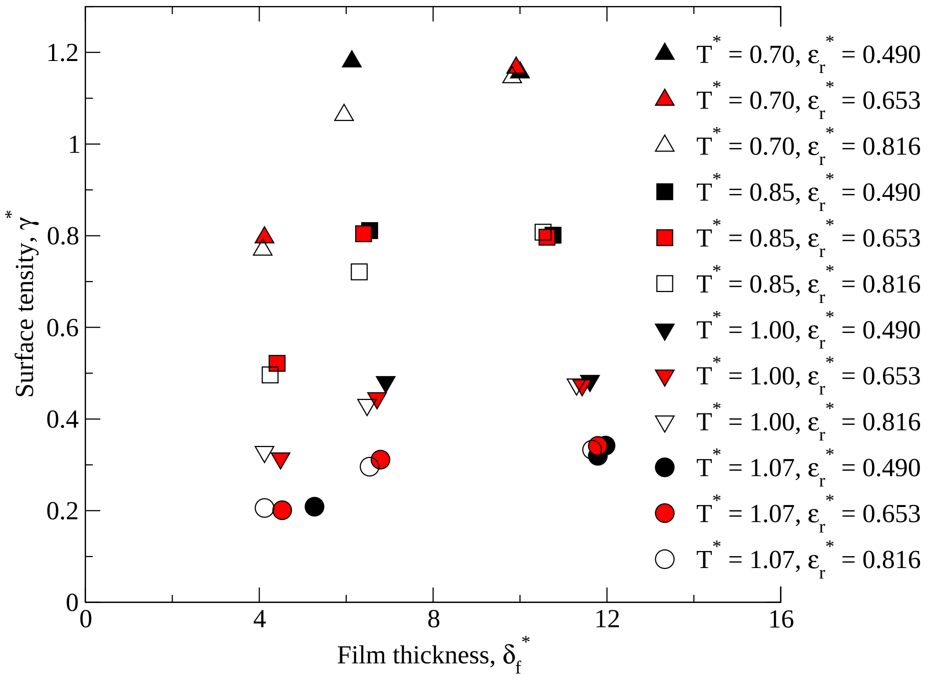


Figure 3.11: Variation of surface tension with the film thickness, δ_f. $\sigma_{sf} = 0.863436\sigma$.

CHAPTER 4

CONTACT ANGLE OF LIQUID DROPLET

4.1 Contact Angle

A liquid droplet adjacent to a solid surface has two equilibrium regimes: a partial wetting (with a finite contact angle $\theta > 0$) and a complete wetting ($\theta = 0$). In the former case, three phases are in contact at the line as in Fig. 1.1. Each interface has certain associated energy per unit area, γ_{sl}, γ_{sv} and γ_{lv} where subscripts s, l and v correspond to solid, liquid and vapor phases respectively. Using the force balance at the contact line or minimizing the equilibrium energy, one obtains the famous Young's equation

$$\gamma_{sv} = \gamma_{sl} + \gamma_{lv} \cos\theta. \tag{4.1}$$

or

$$\cos\theta = \frac{\gamma_{sv} - \gamma_{sl}}{\gamma_{lv}} \tag{4.2}$$

For $|\gamma_{sv} - \gamma_{sl}| < \gamma_{lv}$, we get partial wetting condition and a contact angle between $0°$ and $180°$ is obtained.

4.2 Droplet Simulation

The contact angle of Lennard-Jone liquid droplet is simulated on a solid surface. The solid-fluid interaction is modeled by a (9,3) potential assuming that the solid surface is semi-infinite as discussed in the section 2.3.1. The interaction potential between a solid wall and a Lennard-Jones atom which is at distance z from the solid wall is given as

$$\psi(z) = \frac{2\sqrt{2}\pi\epsilon_{sf}\sigma_{sf}^3}{45R_0^3}\left[2\left(\frac{\sigma_{sf}}{z}\right)^9 - 15\left(\frac{\sigma_{sf}}{z}\right)^3\right]. \tag{4.3}$$

The lattice constant of an FCC(111) solid surface is R_0 as given in Eq. (2.6). The force on the fluid atom due to the solid wall is equal to $-\psi^{'}(z)$ and is given by

$$F_s(z) = \frac{2\sqrt{2}\pi\epsilon_{sf}\sigma_{sf}^3}{5R_0^3 z}\left[2\left(\frac{\sigma_{sf}}{z}\right)^9 - 5\left(\frac{\sigma_{sf}}{z}\right)^3\right]. \tag{4.4}$$

The minimum of $\psi(z)$ is obtained by $d\psi/dz = 0$ which gives

$$z = \left(\frac{2}{5}\right)^{1/6}\sigma_{sf} \tag{4.5}$$

and the minimum $\psi(z)$ is given by

$$\psi_{min} = -\frac{4\sqrt{5}\pi}{9}\left(\frac{\epsilon_{sf}\sigma_{sf}^3}{R_0^3}\right). \tag{4.6}$$

Molecular dynamics simulations are carried out to study the contact angle dependence on the solid-liquid strength and the system temperature. One thousand atoms are simulated in a domain of $30 \times 30 \times 30\sigma^3$. The solid potential $\psi(z)$ is applied at the $z = 0$ (bottom surface). At the top surface, a mirror boundary condition is applied. Any atom which reaches the top boundary, continues with the same x and y velocity, but reverses its z velocity. The system has doubly periodic boundary conditions in the x and y direction. The velocity of an atom crossing the x and y boundary remains unchanged. Atoms leaving the $x = L_x/2$ boundary enter the domain at $x = -L_x/2$ and vice versa. In the beginning of the simulation, atoms are stacked in a simple cubic structure near the bottom surface as shown in Fig. 4.1. Simulation domain is shown in wire-frame as $-15\sigma \leq x, y, z \leq 15\sigma$. The bottom layer of the fluid atoms are placed σ_{sf} from the solid wall ($z = 0$). Each atom is spaced 1.13σ from its neighboring atoms in all three directions. Initially, velocities are assigned to the atoms randomly such that $\sum_{i=1}^{N_{atm}} \frac{1}{2} m v_i^2 = \frac{3}{2} k_B T N_{atm}$ where m is the mass of the atom and T is the desired system temperature.

The Verlet algorithm Eq. (2.2, 2.3) is used to integrate the motion of the atoms. Berendsen's thermostat is applied at every time step for the first 100,000 time-steps by multiplying the velocity components by factor α, where α is given as

$$\alpha = \sqrt{1 + \frac{\Delta t}{\tau} \left(\frac{T}{T_{current}} - 1 \right)}, \tag{4.7}$$

and τ is a rescaling parameter. In this case, $\tau = 20\Delta t$. $T_{current}$ is the current system temperature given by

$$T_{current} = \frac{1}{3 k_B N_{atm}} \sum_{i=1}^{N_{atm}} m v_i^2. \tag{4.8}$$

As the system is under the influence of a solid wall, the above thermostat is used every 5th time step for the next 1.05 million time steps to keep the system at the desired temperature. The center of mass of the system is adjusted in the $x - y$ plane after every 500 time-steps without altering the relative positions of the atoms. It helps in accurate density evaluation. During these simulations

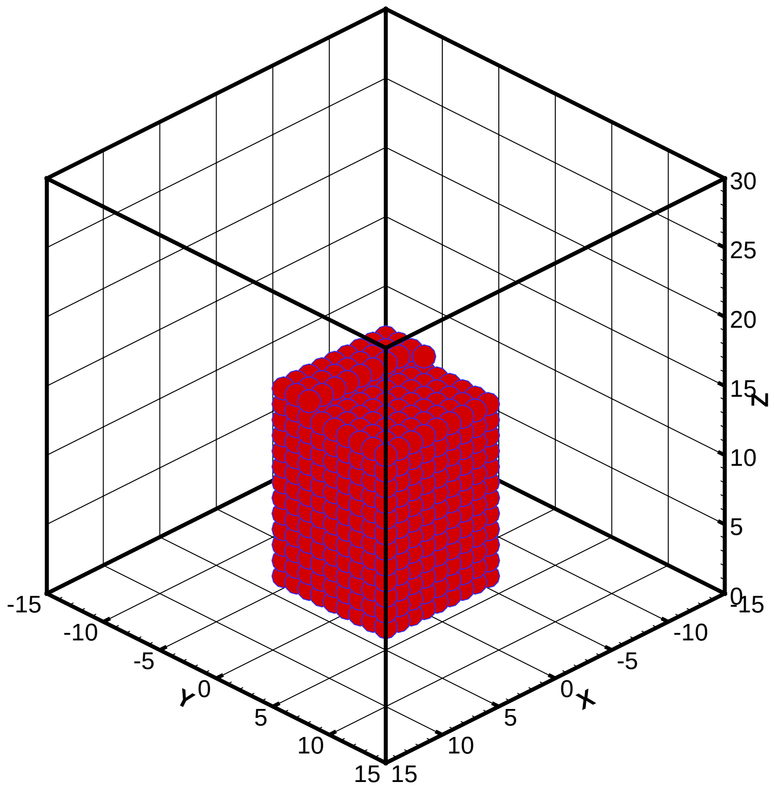


Figure 4.1: Initial configuration for the liquid droplet simulation

a non-dimensional time-step of 0.005 is used. Statistics are sampled after 150k time-steps at every 25 time-steps. Densities are sampled in the lower 1/8 domain ($0 \leq x, y \leq 15\sigma$ and $0 \leq z \leq 15\sigma$) using a $150 \times 150 \times 150$ sampling box.

To investigate the contact angle, densities are post processed in two center planes: $x - z$ plane ($0 \leq y \leq 15\sigma$) and $y - z$ plane ($0 \leq x \leq 15\sigma$). Densities are averaged over a 0.3σ thick center plane. A contour line corresponding to the average of liquid and vapor densities is plotted and it defines the boundary of the liquid droplet as shown in Fig. 4.2. The solid surface is applied at $z = 0$. The droplet boundary shows statistical fluctuations and has a curved interface. The contact angle is calculated by drawing a tangent line (A-B) near the contact line such that line A-B overlaps the contour in $0 \le z \le 3\sigma$ region with good match. The co-ordinates of points A and B give the slope of the tangent line and hence the contact angle. For each test case, the contact angle is measured in the $x - z$ and $y - z$ planes.

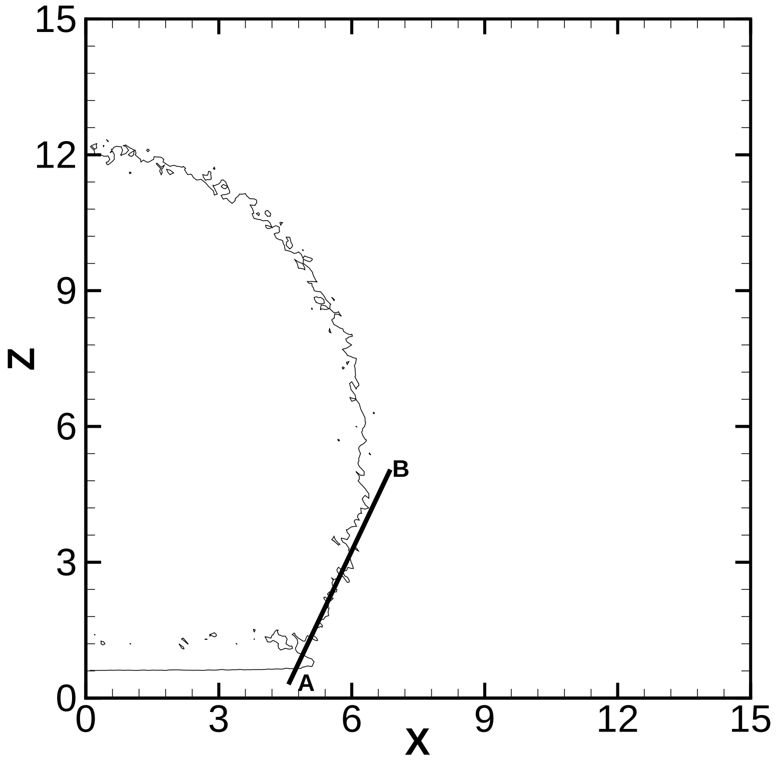


Figure 4.2: Liquid droplet and the contact angle.

We investigated contact angle variation with temperature and the solid-fluid potential parameters. Simulations are carried at $\sigma_{sf} = 0.80$ and 0.863436. $\sigma_{sf} = 0.863436$ corresponds to the Argon-Platinum interaction as $\sigma_{sf} = (\sigma_{ss} + \sigma_{ff})/2.0$.

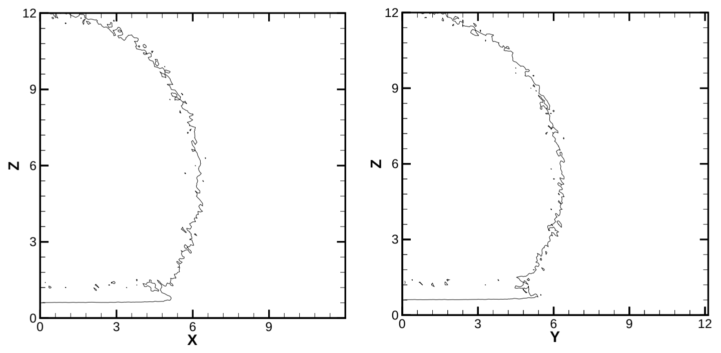
For $\sigma_{sf}/\sigma = 0.8$, we simulated a droplet for $\epsilon_r = \epsilon_{sf}/\epsilon = 0.408$, 0.490, 0.572, 0.653, 0.735, 0.817 and for non-dimensional temperatures from 0.70 to 1.07. At

$T^* = 0.70$, the liquid droplet profile is shown in Fig. 4.3 and where $15\sigma \times 15\sigma$ region of $x - z$ and $y - z$ plane is cropped for plotting.

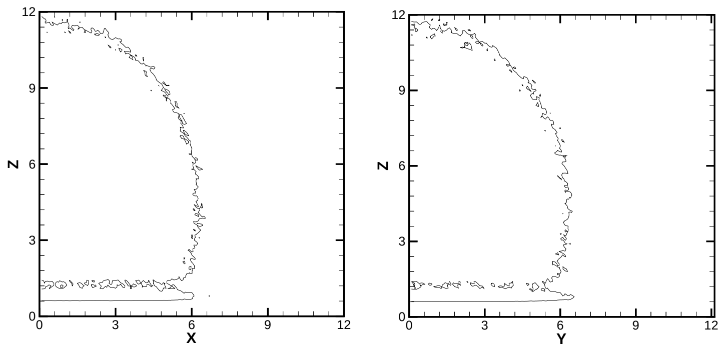
From the droplet contours, we observe that it has a spherical shape. The contact angle can also be calculated by fitting a circular profile with the droplet contour and evaluating the tangent line at $z = 0$, however a tangent line near the contact line works equally well and has been used for the measurement. We can also observe layering adjacent to the solid wall which suggests that the liquid forms a quasi crystaline monolayer near the solid surface. The contour line corresponds to $\rho^* = 0.40$. As ϵ_r is increased from Fig. 4.3(a) to Fig. 4.3(c), we notice that layering gets stronger which causes a depletion of the density adjacent to the monolayer. It is clearly noticeable in the Fig. 4.3(c) where the density is below 0.4 and shows a breakup of the liquid region. The solid-liquid energy parameter affects the droplet base and height. Higher ϵ_r spreads the droplet and has lower droplet height. From the Young's equation, it can be inferred that the difference between γ_{sv} and γ_{sl} increases by increasing ϵ_r as the liquid monolayer formation near the solid surface affects this term and the contact angle decreases.

At temperature $T^* = 1.0$, the droplet contours are shown for $\epsilon_r = 0.408$, 0.490 and 0.572 in Fig. 4.4(a), (b) and (c) respectively. At higher temperature, the vapor density increases. Higher statistical fluctuations are also observed near the liquid-vapor interface region. However the trend is similar to that at the lower temperatures. The droplet spreads at higher solid-liquid strength and shows lower contact angle.

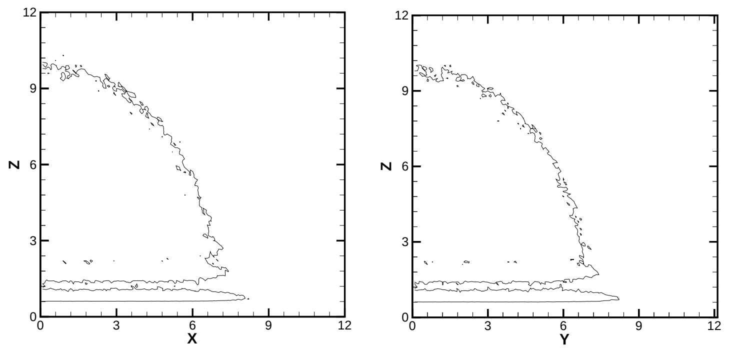
From the droplet contour, the contact angles are evaluated in both $x - z$ and $y - z$ planes and is averaged. The difference between the two contact angles measured in two center planes is around couple of degrees. The contact angle of the droplet is plotted in Fig. 4.5 (a) at various temperatures and the solid-liquid



(a)

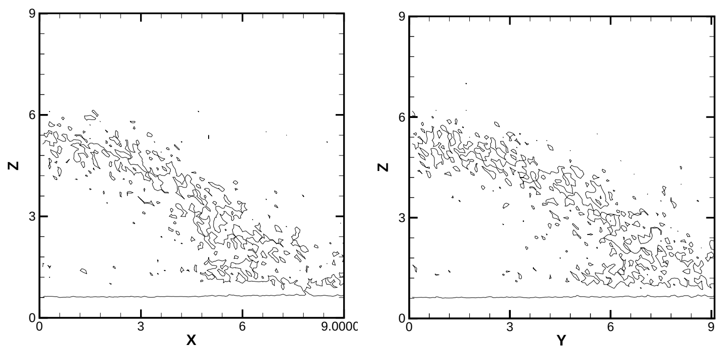


(b)

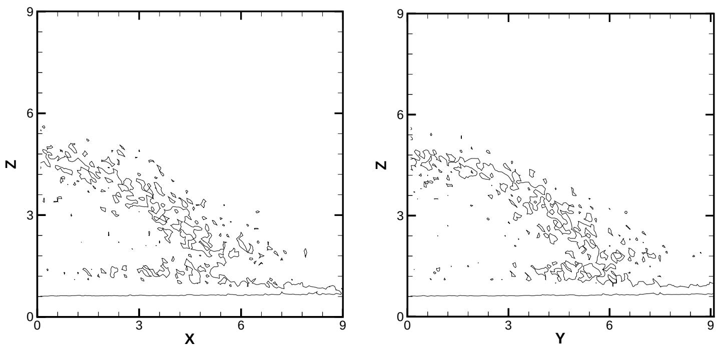


(c)

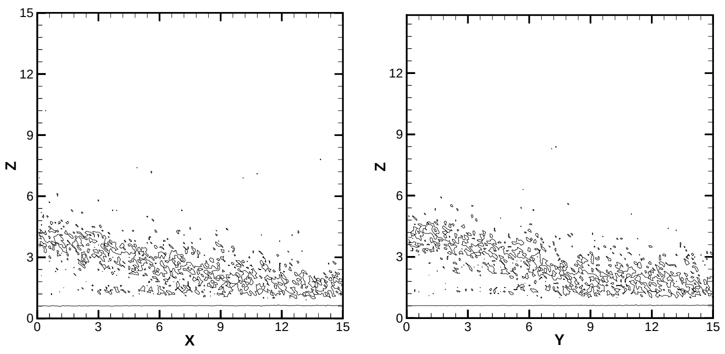
Figure 4.3: Droplet contours at $T^* = 0.7$. (a) $\epsilon_r = 0.408$, (b) $\epsilon_r = 0.572$ and (c) $\epsilon_r = 0.817$. Left one is center xz plane and right one is center yz plane.



(a)



(b)



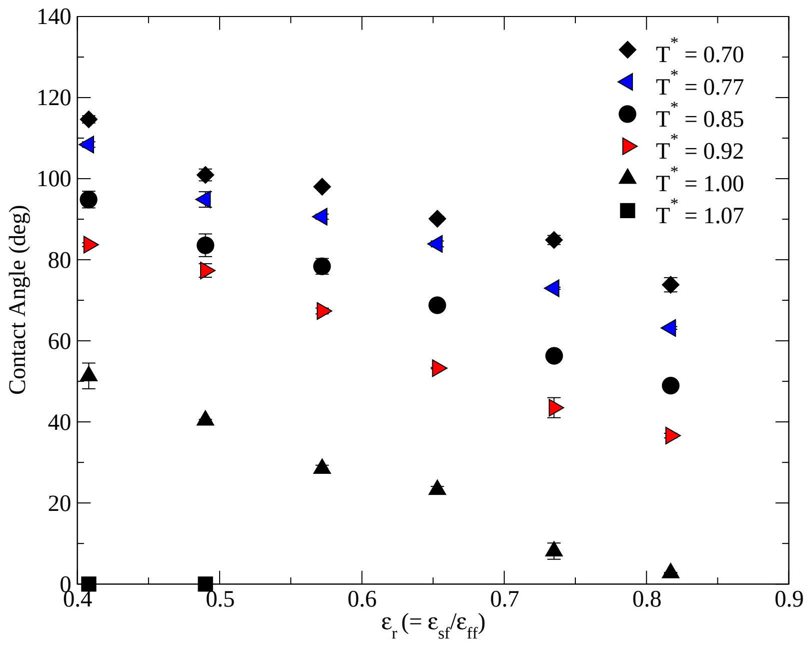
(c)

Figure 4.4: Droplet contour at $T^{*} = 1.0$. (a) $\epsilon_r = 0.408$, (b) $\epsilon_r = 0.490$ and (c) $\epsilon_r = 0.572$. Left one is center xz plane and right one is center yz plane.

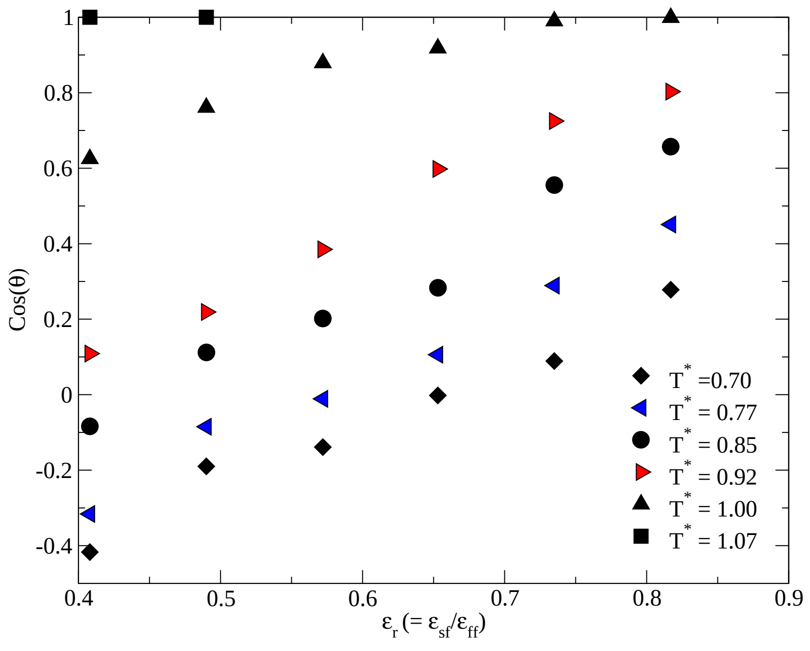
energy parameters. Fig. 4.5 (b) plots the variation of the cosine of the contact angle. The error bar is shown in the contact angle corresponds to the difference in the contact angle measurement in two different center planes. Increasing the value of ϵ_r decreases the contact angle at a given system temperature. Macroscopically, it suggests that increasing ϵ_r increases the $\gamma_{sv} - \gamma_{sl}$. At lower temperatures and lower ϵ_r, $\gamma_{sv} < \gamma_{sl}$ and we observe the contact angles more that $90°$. A contact angle of $90°$ corresponds to $\gamma_{sv} = \gamma_{sl}$.

If one fixes the solid-fluid combination (ϵ_r), and gradually increases the temperature, the contact angle decreases as shown in Fig. 4.6 for the various solid-liquid interaction strengths. The contact angle decreases slowly in the temperature range of 0.7 to 0.9. For $T^* \geq 0.9$, the contact angle decays rapidly and the solid gets fully wet at $T^* = 1.0 \sim 1.1$, which is lower than the critical temperature ($T_c^* = 1.26$). The Interfacial tension γ_{lv} decreases with temperature as shown in the previous chapter. From the macroscopic Young's equation, it can be inferred that γ_{lv} approaches the difference between γ_{sv} and γ_{sl} and a wetting condition is observed. The contact angle of the high energy surface (high ϵ_r) shows lower contact angle at all temperatures and also approaches the wetting condition at lower temperature.

Sullivan [2] used the van der Waals model to study the contact angle dependence on temperature. He concluded that the contact angle depends on the T/T_c and ϵ_r/kT_c and showed a similar trend of the contact angle dependence on temperature for given ϵ_r. Adamson [3] used a potential distortion model to study the variation of the contact angle and validated for n-decane on teflon surface and n-octane on teflon surface. For the n-octane on teflon surface, they found that the contact angle reaches $0°$ when temperature is 0.78 of the bulk critical temperature whereas in the n-decane on teflon surface, this was given as 1.24 times



(a)



(b)

Figure 4.5: Variation of (a) the contact angle and (b) $\cos(\theta)$ with ϵ_r, ($\sigma_{sf} = 0.8\sigma$).

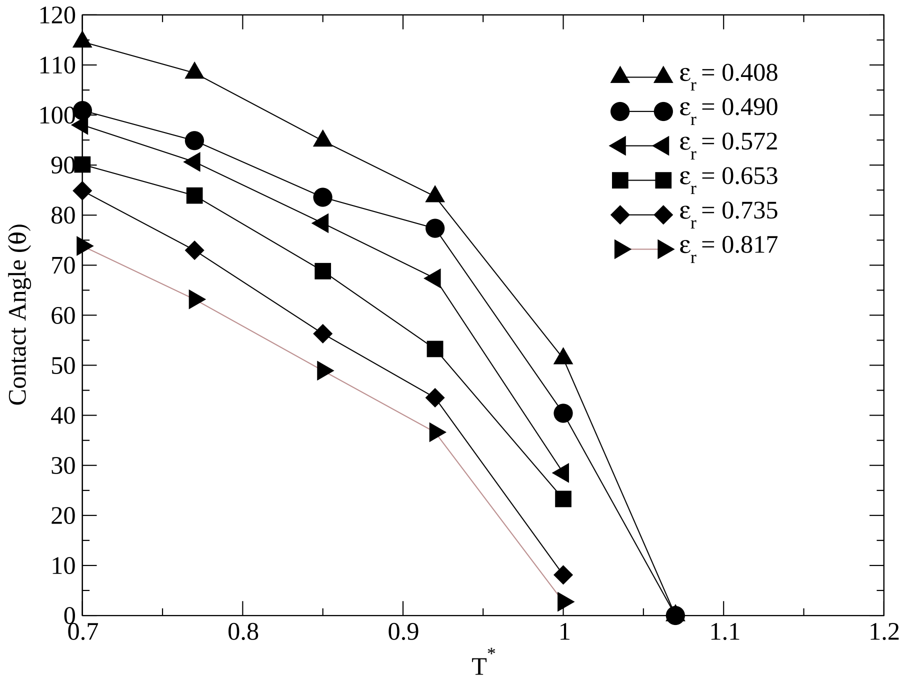


Figure 4.6: Variation of the contact angle with temperature ($\sigma_{sf} = 0.8\sigma$)

the normal boiling point, which is lower than the critical bulk temperature. This variation is shown in Fig. 4.8(a). Lay and Dhir [4] (Fig. 4.8(b)) carried out a theoretical study of the variation of the contact angle with the saturated pressure and temperature for water and found that it decays rapidly at higher temperatures. These studies were for different test fluids but the contact angle variation with the temperature shows similar trend.

Solid-fluid interaction is also dependent on the length parameter σ_{sf}. In the above results, $\sigma_{sf} = 0.8\sigma$ was employed. To investigate the effect of σ_{sf} on contact angle, σ_{sf} is increased from 0.8σ to 0.863436σ. Argon-platinum interaction length parameter σ_{sf} corresponds to 0.863436σ. Liquid droplet is simulated for $\sigma_{sf} = 0.863436\sigma$ and ϵ_r and T^* is varied. The contact angle is evaluated as before and shown in Fig. 4.9. The contact angle decreases with increasing ϵ_r as observed for

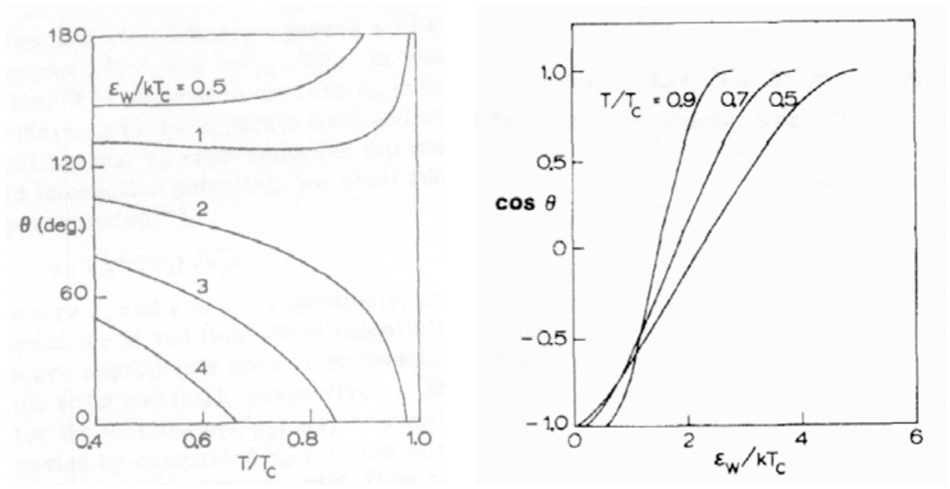
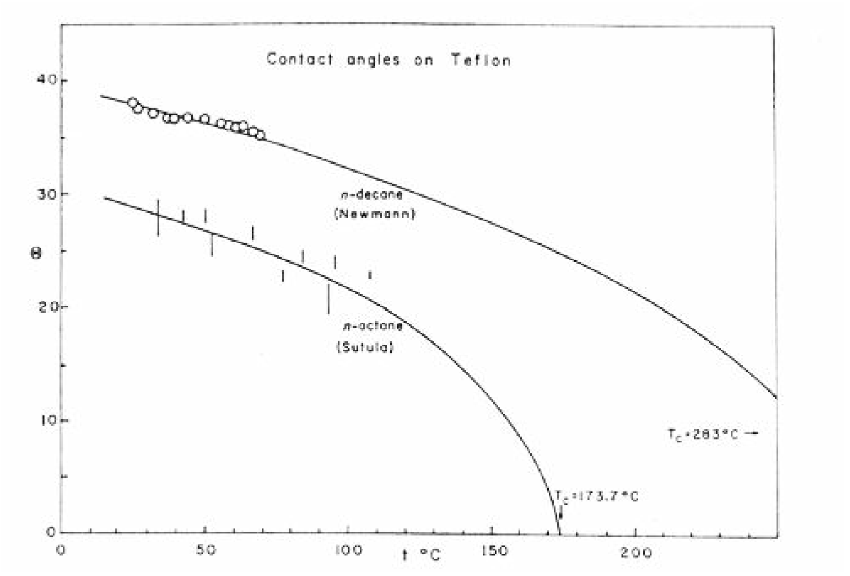


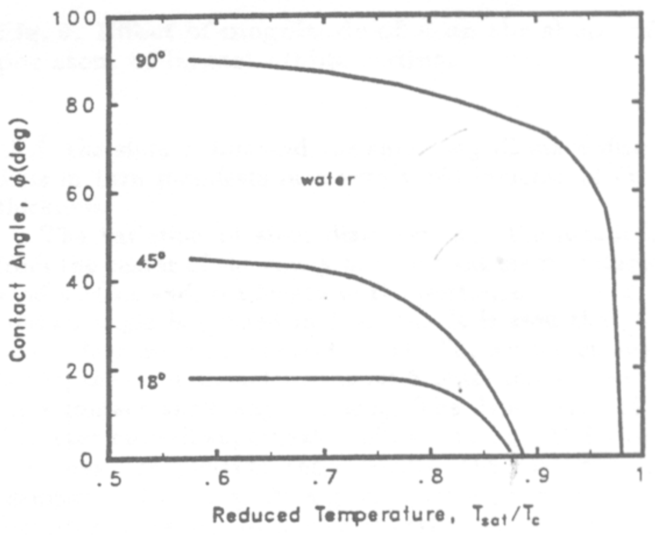
Figure 4.7: Variation of (a) α and (b) β with temperature. (From Sullivan [2])

the $\sigma_{sf} = 0.8\sigma$ case, however the contact angle is smaller. For $\epsilon_r \geq 0.490$ and $T^* \geq 0.70$, the contact angles are less than 90°. Variation of the contact angle with temperature is shown in Fig. 4.10. The contact angle decreases with increasing temperature and decays rapidly after $T^* = 0.90$ as γ_{lv} decreases with temperature. At higher $\epsilon_r (= 0.817)$, the contact angle decays linearly and approaches the wetting condition at $T^* = 0.94$.

Liquid-vapor interfacial tension decreseas with the temperature and $\cos(\theta)$ increases with the temperature for all the above cases. Macroscopically, the difference between the solid-vapor and the solid-liquid surface tension can be evaluated from the Young's equation as $\gamma_{sv} - \gamma_{sl} = \cos(\theta)\gamma_{lv}$. This is plotted in Fig. 4.11. Fig. 4.11(a) shows the variation for $\sigma_{sf} = 0.80\sigma$ and Fig. 4.11(b) shows the variation for $\sigma_{sf} = 0.863436\sigma$. At lower ϵ_r, this difference has negative value at lower system temperature and $\sigma_{sf} = 0.8\sigma$, but goes to the positive value for $\sigma_{sf} = 0.863436\sigma$. Negative difference implies that $\gamma_{sl} > \gamma_{sv}$. As temperature increases, $\gamma_{sv} - \gamma_{sl}$ gradually increases for lower ϵ_r. At higher ϵ_r, it shows a different trend. The difference between γ_{sv} and γ_{sl} increases with temperature

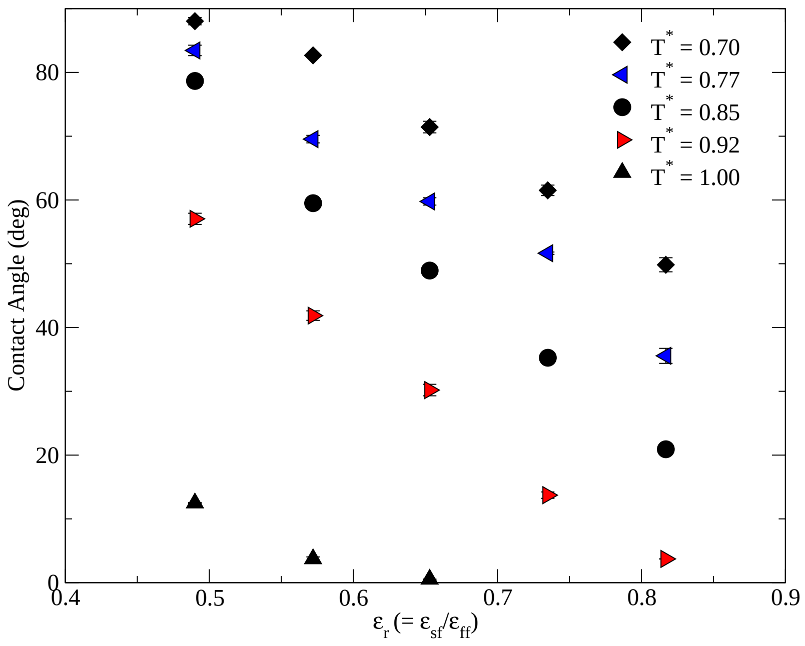


(a)

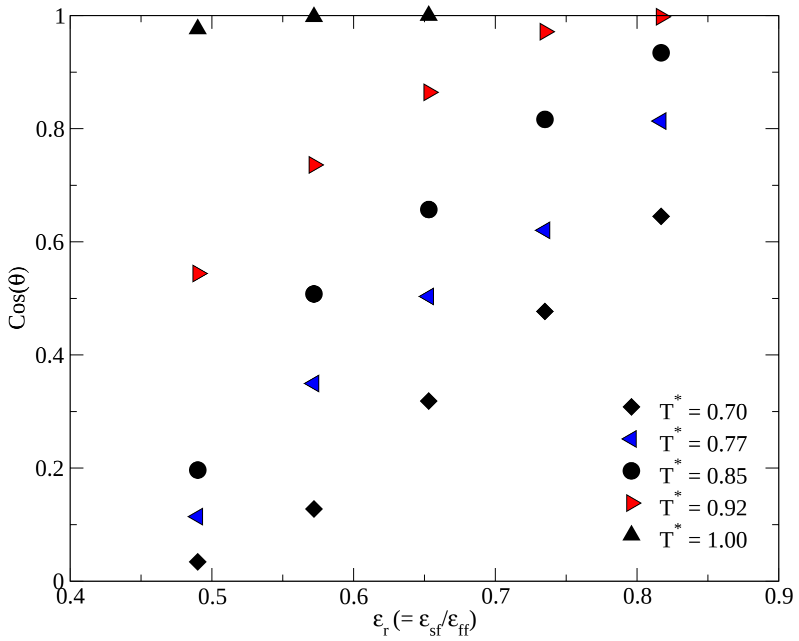


(b)

Figure 4.8: Contact angle variation with temperature for (a) the organic fluids (From Adamson [3]) (b) the water (From Lay and Dhir [4])



(a)



(b)

Figure 4.9: Variation of (a) the contact angle and (b) $\cos(\theta)$ with ϵ_r, ($\sigma_{sf} = 0.863436\sigma$).

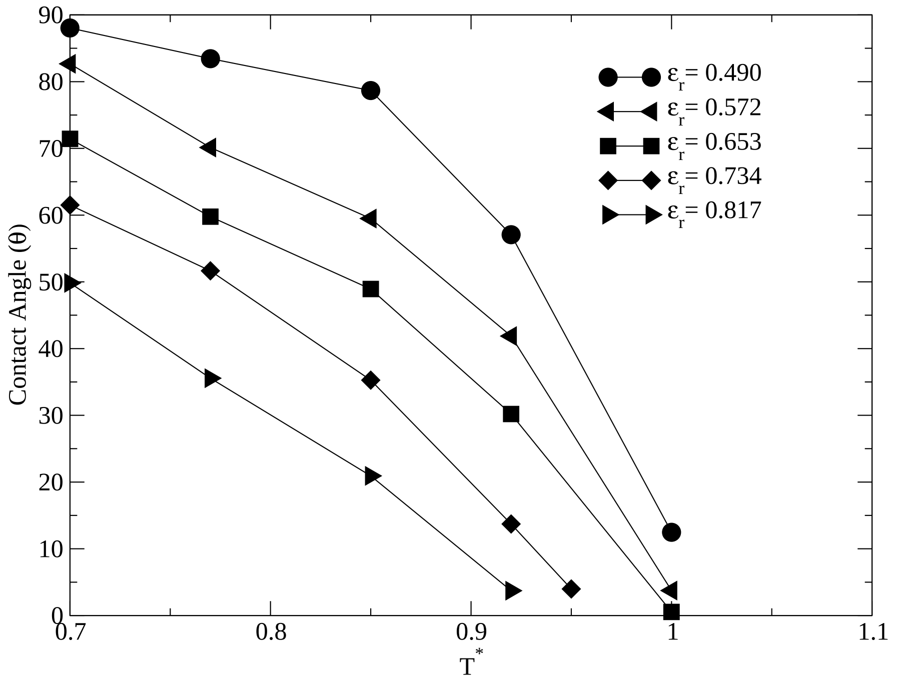
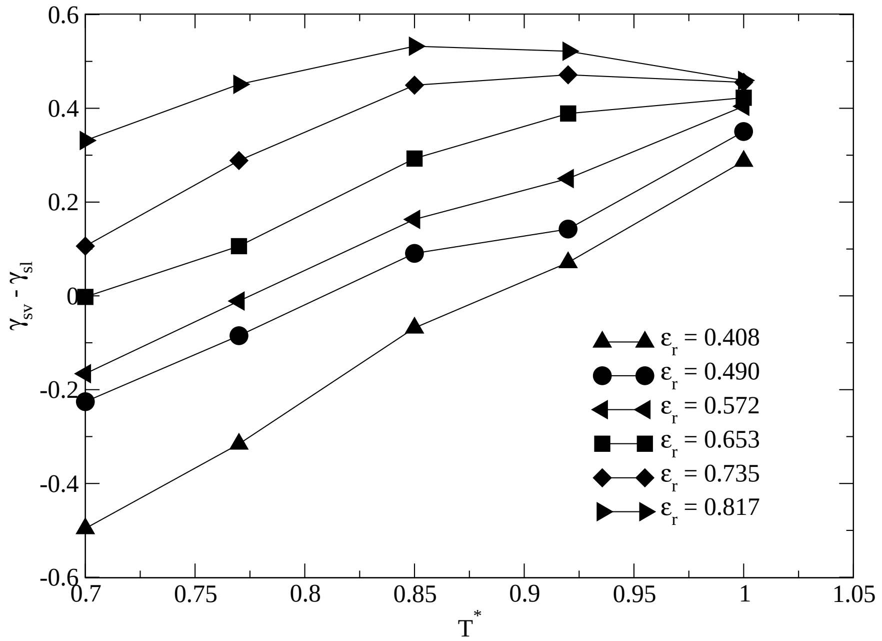


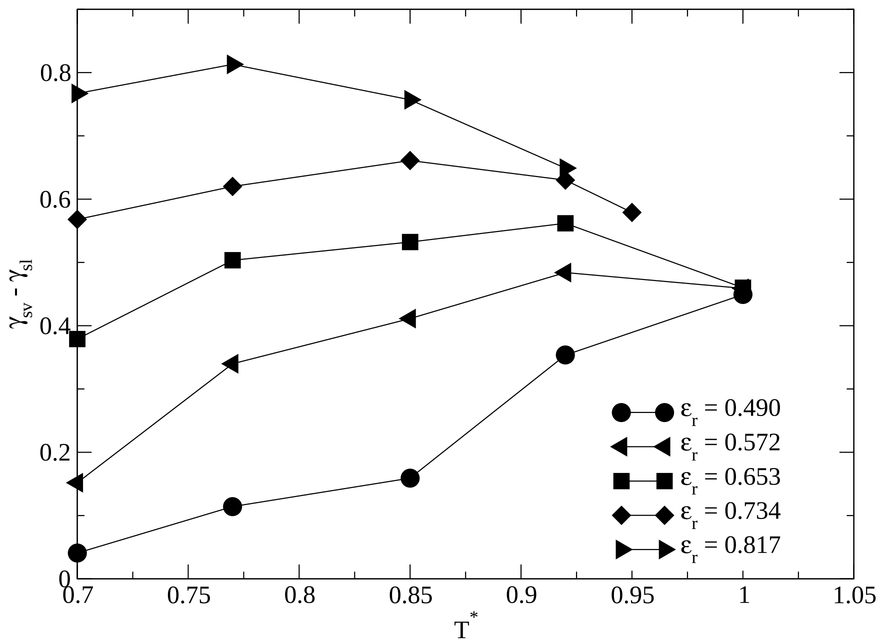
Figure 4.10: Variation of the contact angle with temperature ($\sigma_{sf} = 0.863436\sigma$).

and reaches maximum. As system temperature is increased further, this difference starts decreasing with temperature as the LJ fluid starts wetting the solid surface. In Chapter 3, we looked at the variation of the liquid-vapor interfacial tension and found that it is weakly dependent on the solid-fluid interaction strength for the uniform films. On the other hand, $\gamma_{sv} - \gamma_{sl}$ shows strong dependence on the ϵ_r and the σ_{sf} For thinner films, the liquid-vapor surface tension is lower than the macroscopic value as observed in Table 3.3 and the difference between γ_{sv} and γ_{sl} would decrease further.

Based on the above simulations, the contact angle is found to decrease with increasing ϵ_r, σ_{sf} and temperature. The parameters ϵ_r and σ_{sf} are related through ψ_{min} in Eq. (4.6). Maruyama *et al.* [35] used a (10,4) potential for ψ (one layer of solid atoms) and observed that the contact angle depends on ψ_{min} at a given



(a)



(b)

Figure 4.11: Variation of $\gamma_{sv} - \gamma_{sl}$ with temperature for (a) $\sigma_{sf} = 0.80\sigma$ and (b) $\sigma_{sf} = 0.863436\sigma$).

temperature. A solid potential Eq. (4.3) is used in our case, and the contact angle dependence on ψ_{min} (Eq. (4.6)) is investigated. Table 4.1 shows values of ψ_{min} for combinations of ϵ_r and σ_{sf} used in the simulations.

Table 4.1: Value of ψ_{min} used in the simulations.

ϵ_r/ϵ	σ_{sf}/σ	ψ_{min}/ϵ
0.408	0.8	-1.207
0.490	0.8	-1.448
0.572	0.8	-1.690
0.653	0.8	-1.931
0.735	0.8	-2.172
0.817	0.8	-2.414
0.490	0.863436	-1.821
0.572	0.863436	-2.124
0.653	0.863436	-2.428
0.735	0.863436	-2.731
0.817	0.863436	-3.035

The variation of contact angle with temperature for various ψ_{min} is plotted in Fig. 4.12. Contact angle variation is shown for ψ_{min} ranging from -3.035ϵ to -1.207ϵ. For each case, the contact angle decreases with increasing temperature which is not linear. However the contact angle variation with the ϵ_r can be approximated as linear profile. As ψ_{min} is directly proportional to the ϵ_r, the contact angle varies linearly with ψ_{min}. From Fig. 4.12, it is noted that same ψ_{min} has the same contact angle. This can be observed for $\psi_{min} = -2.124\epsilon$ (i.e. $\epsilon_r = 0.572\epsilon$ and $\sigma_{sf} = 0.863436\sigma$) and $\psi_{min} = -2.172\epsilon$ (i.e. $\epsilon_r = 0.735\epsilon$ and $\sigma_{sf} = 0.8\sigma$) show the same contact angles. Also $\psi_{min} = -2.414\epsilon$ (i.e. $\epsilon_r = 0.817\epsilon$ and $\sigma_{sf} = 0.8\sigma$) and $\psi_{min} = -2.428\epsilon$ (i.e. $\epsilon_r = 0.653\epsilon$ and $\sigma_{sf} = 0.863436\sigma$) show the same contact angles. This means that the depth of the potential well determines the contact angle irrespective of the energy and length parameters in the solid-fluid potential.

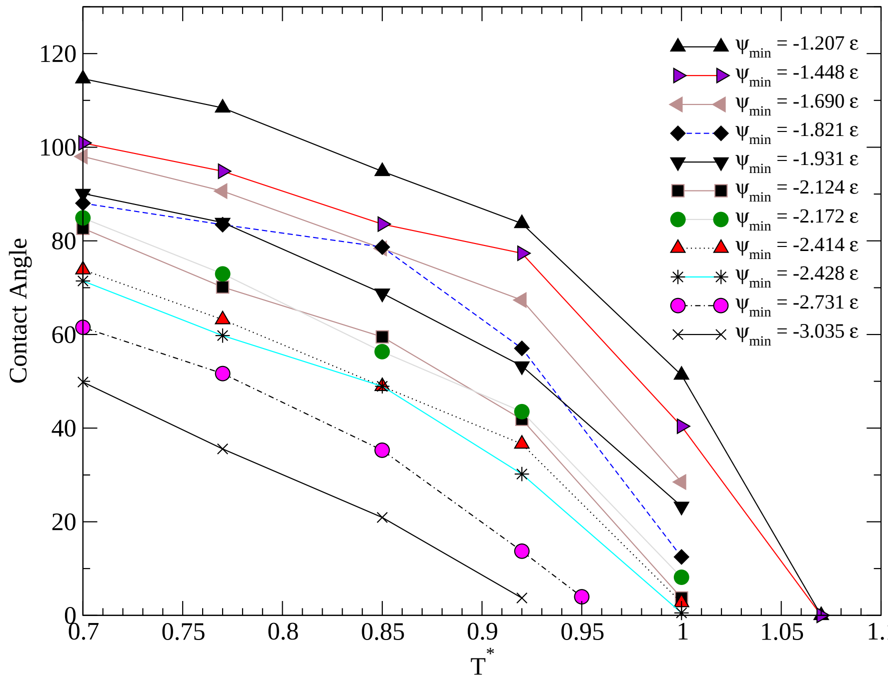


Figure 4.12: Variation of the contact angle with temperature for different ψ_{min}

The contact angle variation with temperature and ψ_{min} is investigated. The contact angle varies linearly with ψ_{min} for a given temperature and can be expressed as

$$\theta = \alpha(T^*) + \beta(T^*)\psi^*_{min}, \tag{4.9}$$

where α and β are functions of temperature T^* and ψ^*_{min} which is non-dimensionalized with ϵ. From the slope of the variation of the contact angle with ψ, we get the variation of α and β with temperature as shown in Fig. 4.13 (a) and (b) respectively. The variation of α and β is fitted with second order polynomial as

$$\alpha = -852T^{*2} + 1274.54T^* - 322.6, \tag{4.10}$$

and

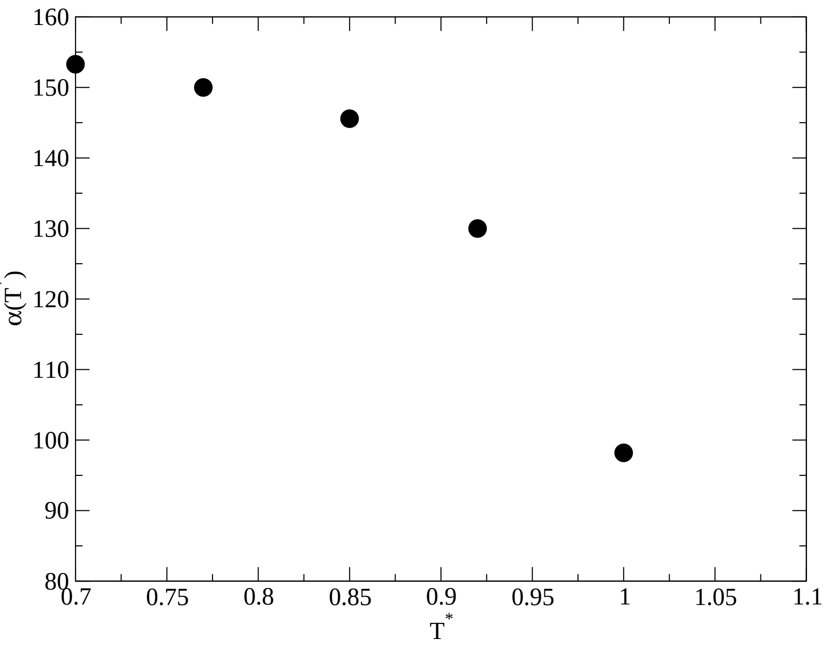
$$\beta = -101.75T^{*2} + 199.11T^{*} - 56.149. \tag{4.11}$$

Eq. (4.9) with Eq. (4.10) and Eq. (4.11) gives the contact angle of the Lennard-Jones fluid for a wide variety of the solid-fluid interactions. The contact angles from the above correlation is compared with all the simulation cases in Fig. 4.14. Predicted contact angles show good match with our simulation results except near the complete wetting condition ($\theta \sim 0^{\circ}$) as there is significant error in contact angle determination from the density profile. Contact angle of argon droplet can be predicted for the various solid surfaces using Eq. (4.9).

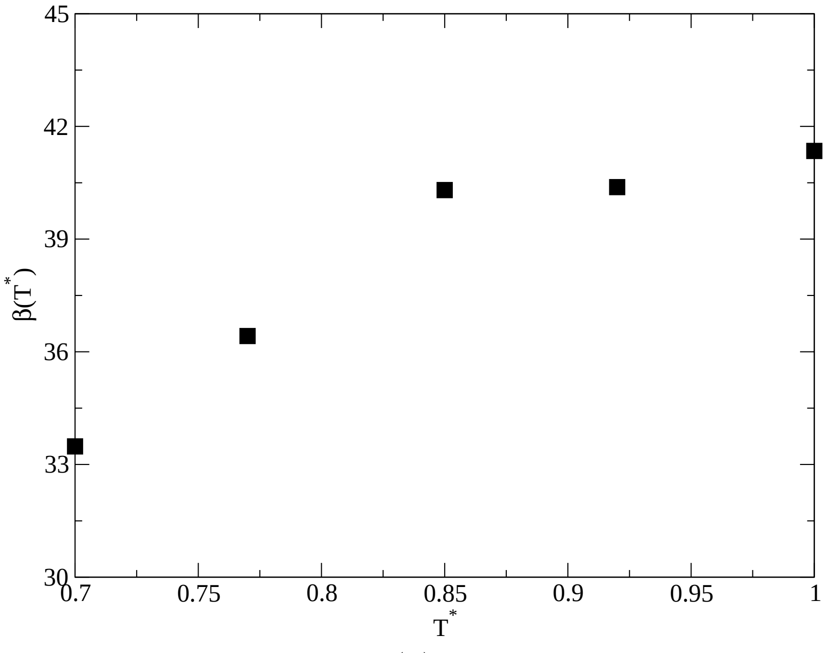
The Hamaker constant (A) is defined in terms of the solid-fluid interaction in Eq. (2.58). Hamaker constant quantifies the solid-liquid interaction and can be represented in terms of ψ_{min} as

$$A = -\frac{9\sqrt{2}}{\sqrt{5}}\psi_{min}\rho_{liq}\sigma_{sf}^{3}. \tag{4.12}$$

In the above equation, we notice that the Hamaker constant depends on the fluid density, σ_{sf} and ψ_{min}. The contact angle dependence on the Hamaker constant is investigated for two different σ_{sf}. The non-dimensional temperature is varied from 0.7 to 1.0. The variation of contact angle with Hamaker constant is shown in Fig. 4.15(a). Hamaker constants of the solid-fluid combinations are varied from 10^{-20}J to 5×10^{-20}J. The contact angle decreases with increasing A. Solid and open symbols correspond to the $\sigma_{sf} = 0.8\sigma$ and $\sigma_{sf} = 0.863436\sigma$ respectively. We notice that contact angle variation with temperature does not concide for two different σ_{sf}. Thus the contact angle is not uniquely defined for a given Hamaker constant and temperature. The contact angle variation with $A/\rho_{liq}\sigma_{sf}^{3}$ is plotted in Fig. 4.15(b). It is observed that solid and open symbols align at a given temperature. Term $A/\rho_{liq}\sigma_{sf}^3$ is proportional to ψ_{min} from Eq. (4.12) and so the contact angle is uniquely defined for ψ_{min}.



(a)



(b)

Figure 4.13: Variation of (a) α and (b) β with temperature.

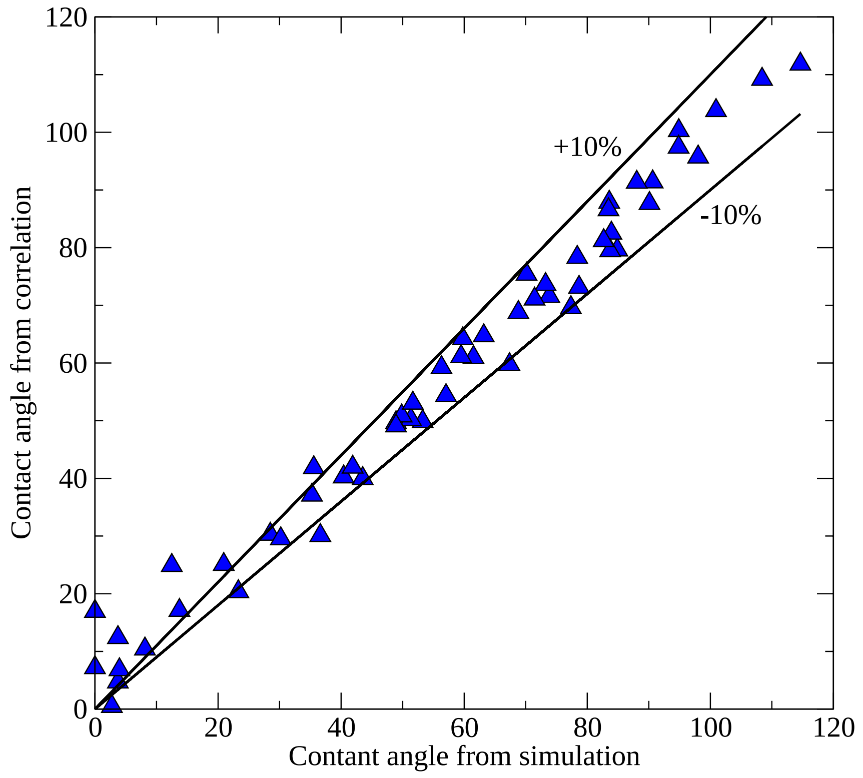
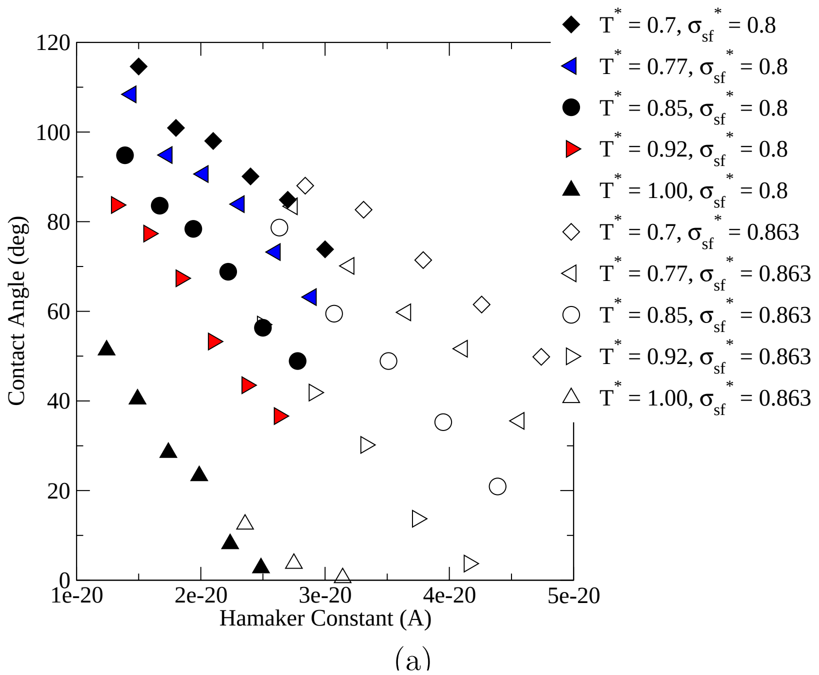


Figure 4.14: Comparison of the contact angle predicted from the correlation and simulation data's.



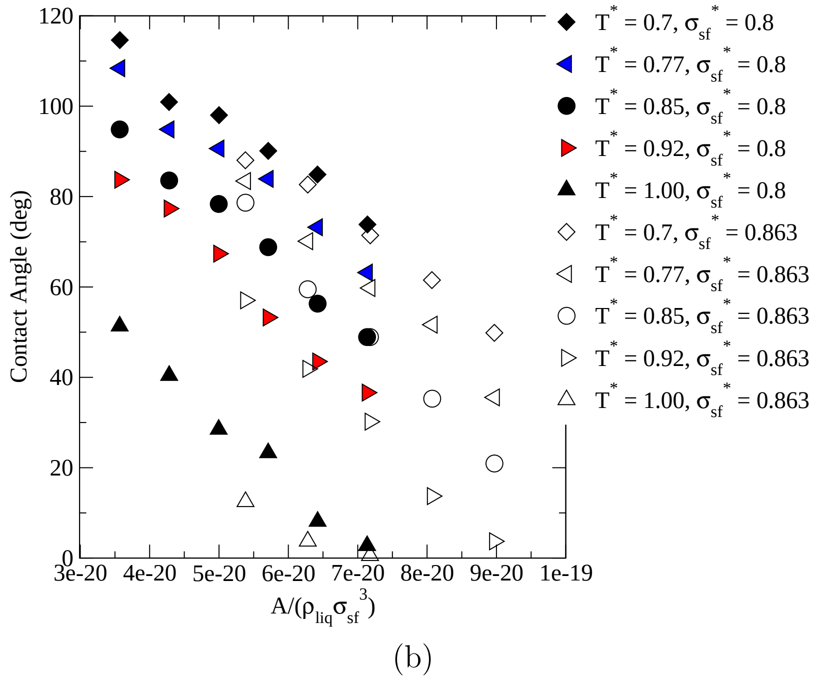


Figure 4.15: Contact angle dependence on (a) Hamaker constant, A and (b) $A/(\rho_{liq}\sigma_{sf}^3)$.

CHAPTER 5

IMPLEMENTATION OF P^{3}M METHOD FOR FORCES AND SURFACE TENSION

5.1 P^{3}M Method

Evaluation of forces/energies is an important step of molecular simulation as forces are used in the integration of Newton's equation for tracking the atomic motion. A neighbor list method with a finite cutoff radius provides a simpler way to evaluate the forces, but a slowly decaying term or short cutoff radius can significantly increase the error in evaluation. Due to computational restrictions ($\mathcal{O}(N^3)$), it is not possible to include an infinite cutoff radius in the neighbor list method. The famous *Ewald sum* (Ewald [44], de Leeuw *et al.*[45, 46]) for the coulomb potential ($1/r$) splits the potential into two sums which converge well. It avoids large errors which may appear due to the finite cutoff for coulomb potential (Deserno and Holm [47]). This approach is still too time consuming for large number of particles and various alternative algorithms have been devised to speed up the computation using fast Fourier transforms (FFT). Particle-particle/particle-mesh (PPPM or P^{3}M) method (Hockney and Eastwood [53]) is one of these hybrid algorithms which uses an FFT accelerator to speedup the computation and minimize the overall error. However this method is not used for the evaluation of the dispersive force and the surface tension in the literature.

We implemented and tested it for Lennard-Jones system.

P^{3}M method enables long range forces to be evaluated effectively and accurately for a large system. The basic idea is to split the required term as the sum of two parts.

$$\mathbf{F}_{ij} = \mathbf{F}_{ij}^{sr} + \mathbf{F}_{ij}^{lr}, \tag{5.1}$$

where the short range part $\mathbf{F}_{ij}^{sr}$ is an exponentially decaying term and has effective compact support for only a few inter-particle separations. This term can be evaluated using direct particle-particle (PP) force summation within a finite cutoff radius similar to the neighbor list method. The long range part $\mathbf{F}_{ij}^{lr}$ is a smoothly varying part and has a Fourier transform which is band-limited and can be represented on mesh. This long range part is non-zero for a limited $\mathbf{k}$ in Fourier space and approximated by particle-mesh (PM) method. So the P^{3}M method falls in between the PP method and the PM method in that it accurately represents the required term for short distance as the PP method and calculates the large distance term as rapidly as the PM method.

5.2 The Short Range Term

The short range term on an atom i is the sum of interparticle short range term due to all other atoms within cutoff r_c. This interparticle force term can be calculated and summed by sweeping all atoms (N) in the system and would have computational expense as $\mathcal{O}(N^2)$, which is not practical for a large number of atoms. This cost is minimized by using a *chaining mesh cell.*

The number of chaining mesh cells in any direction s is the largest integer less than or equal to L_s/r_l, where L_s is the length of the domain in the s direction and r_l is the list radius and it is 15 percent larger than short range cutoff radius

r_c^{sr} . This means that the chaining mesh cell length is always greater or equal to the r_c^{sr}. The idea is similar to the neighbor list method with automatic updates of the neighbor list and we do not have to update the linked list in every cell after every time steps. A linked list of every chaining mesh cell links all atoms in the parent cell. Whenever a linked list is created, we store the positions of all atoms. After every time step, particles new positions are compared with the stored position. If any atom has relative movement more than $0.15 \times r_c$ in any direction, the subroutine is called to create the linked list of each chaining cell and store the positions of all atoms.

To calculate the short range term, we need to sweep over the neighboring atoms stored in an array. This array stores all neighboring atoms within r_l. The linked list is updated as soon as any atom moves by $0.15 \times r_c$, and it guarantees that all atoms within r_c^{sr} are accounted. There is a total of 26 chaining cells around a central cell. To create the neighbor list, 13 adjacent chaining cells are considered along with the current cell because Newton's third law is applicable for both atom i and atom j while calculating the short range term. In our case, we choose the neighbors of chaining cell $\mathbf{q} = (q_1, q_2, q_3)$ to be $(q_1 + 1, q_2 + s, q_3 + t), (q_1, q_2 + 1, q_3 + t), (q_1, q_2, q_3 + 1); s, t = (-1, 0, 1)$. This method costs $\mathcal{O}(13 N_{atm} N_{cell})$ where N_{atm} is number of atoms in the domain and N_{cell} is the number of chaining mesh cells.

5.3 The Long Range Term: Mesh Term

The mesh term is computed using optimized Particle-Mesh (PM) method. We use the charge-potential mesh cell (N_m) for long range computation which is different from the chaining potential mesh cell (N_c) used in the short range calculation. As the short range term decays exponentially within a few interparticle separations, the long range term should be equal to the actual term for larger distance. This is implemented in the following four steps.

1. Charge assignment to charge-potential mesh
2. Solve for the potential
3. Obtain the mesh defined fields
4. Interpolate to obtain the field at the atomic positions

Each step incurs numerical error, however the optimization of the PM method creates inaccuracies in the individual steps which are self canceling overall and have high accuracy.

5.3.1 Charge Assignment

Atoms are considered as unit charged particles (even Lennard-Jones atoms). As long range force needs to be solved in charge-potential mesh, we distribute the charge over the mesh points such that assignments carried over small number of the mesh points, field should tend toward the actual field far from the source and assignment should be smooth with the location of the particle. Computational cost depends on the number of the mesh points for charge assignment and smoothness governs the harmonic content of the calculated field.

Low pass filter is a ideal assignment function, but practically impossible due to computational expense. Three common assignment functions are NGP (nearest grid point, 0th order), CIC(cloud in cell, 1st order) and TSC (triangular shaped cloud, 2nd order) which satisfy progressively higher order long range and smoothness constraint. The lowest order scheme (NGP) gives discontinuous force with location while CIC gives continuous force but discontinuity in the first derivative.

We used TSC assignment function for charge distribution which involves the closest mesh point and twenty six neighboring mesh points. If V_c is the volume of the mesh cell and H_i is the cell length in i^{th} direction, then the *weighting function* is given as

$$W(\mathbf{x}) = \frac{1}{V_c} w\left(\frac{x}{H_1}\right) w\left(\frac{y}{H_2}\right) w\left(\frac{z}{H_3}\right), \tag{5.2}$$

where

$$w(x/H) = \begin{cases} \frac{3}{4} - \left(\frac{x}{H}\right)^2 & |x| \leq \frac{H}{2} \\ \frac{1}{2}\left(\frac{3}{2} - \frac{|x|}{H}\right)^2 & \frac{H}{2} \leq |x| \leq \frac{3H}{2} \\ 0, & \text{otherwise} \end{cases} . \tag{5.3}$$

The particle density $n(x)$ in one dimensional infinite space is given by

$$n(x) = \sum_{i=1}^{N_p} \delta(x - x_i), \tag{5.4}$$

where x_i is the position of particle i. The continuous charge density $\rho^{'}(x)$ for charge q is given by

$$\rho^{'}(x) = \frac{q}{H} \int n(x^{'}) W(x - x^{'}) dx^{'} = \frac{q}{H} n(x) * W(x). \tag{5.5}$$

Mesh defined charge density is obtained by using $x = x_p$ in (5.5) where x_p is the mesh point and $*$ is the convolution function. This can be compactly expressed by sampling function **III** which is an infinite row of δ functions spaced at unit interval.

$$\mathbf{III}(x) = \sum_{n=-\infty}^{\infty} \delta(x - n). \tag{5.6}$$

A generalized mesh defined charge density function $\rho^{\dagger}(x)$ is expressed as

$$\rho^{\dagger}(x) = \mathbf{III}\left(\frac{x}{H}\right)\rho^{'}(x) = \frac{q}{H}\mathbf{III}\left(\frac{x}{H}\right)n(x) * W(x). \tag{5.7}$$

It comprises an infinite row of impulse functions at intervals H and its strength gives the mesh defined charge density. In three dimensions, it is

$$\rho^{\dagger}(\mathbf{x}) = \frac{q}{V_c}\mathbf{III}\left(\frac{\mathbf{x}}{\mathbf{H}}\right)n(\mathbf{x}) * W(\mathbf{x}). \tag{5.8}$$

5.3.2 Potential Solver

Potential on mesh points can be expressed in terms of Green's function G and charge density ρ as

$$\phi^{'}(x) = \int_{-\infty}^{\infty} dx^{'} G(x - x^{'})\rho^{\dagger}(x) \tag{5.9}$$

$$= G^{'}(x) * \rho^{\dagger}(x), \tag{5.10}$$

where prime denotes the continuous functions. In three dimensions, it is expressed as

$$\phi^{'}(\mathbf{x}) = G^{'}(\mathbf{x}) * \rho^{\dagger}(\mathbf{x}). \tag{5.11}$$

The convolution evaluation is computationally expensive. In P^{3}M method, we use fast Fourier transform (FFTs) to evaluate it, which is $\mathcal{O}(N \log N)$. As convolution in real space is similar to product in Fourier space (*Convolution Theorem*) and vice versa, we can compute this in Fourier space as

$$\phi^{'}(\mathbf{x}) \subset \hat{\phi}^{'}(\mathbf{k}) = \hat{G}^{'}(\mathbf{k})\hat{\rho^{\dagger}}(\mathbf{k}). \tag{5.12}$$

The computation is efficiently done using FFTs. At any time step, we know the position of the atoms, which gives the charge density based on the weighting function. The Fourier transform of the charge density is obtained using FFT and multiplied with the Fourier transform of the Green's function. One should note that the Fourier transform of the Green function is obtained only once, in the beginning of the simulation as it only depends on the mesh based domain geometry and evaluation function (like dispersive potential terms) and does not add overhead of evaluating the $\hat{G}$ at every time step.

5.3.3 Mesh Defined Field

The electric field is the gradient of the potential field and can be expressed as two point difference as

$$E^{'}(x) = -\int dx^{'} \left[\frac{\delta(x + H - x^{'}) - \delta(x - H - x^{'})}{2H} \right] \phi^{'}(x^{'}), \tag{5.13}$$

where δ is the Kronecker delta. If we use difference operator as D, the electric field in one dimension becomes

$$E^{'}(x) = -\int dx^{'} \left[D(x - x^{'}) \phi^{'}(x^{'}) \right]. \tag{5.14}$$

The mesh defined field $E^{\dagger}$ is expressed as

$$E^{\dagger}(x) = \mathbf{III}\left(\frac{x}{H}\right) E^{'}(x) = \mathbf{III}\left(\frac{x}{H}\right) \int dx^{'} D(x-x^{'}) \phi^{'}(x^{'}) = \mathbf{III}\left(\frac{x}{H}\right) D(x) * \phi(x). \tag{5.15}$$

Its Fourier transform would give

$$\hat{E}^{\dagger}(k) = -\hat{D}(k)\hat{\phi}(k). \tag{5.16}$$

For multidimensional calculations, it becomes

$$\hat{E}^{\dagger}(\mathbf{k}) = -\hat{D}(\mathbf{k})\hat{\phi}(\mathbf{k}). \tag{5.17}$$

From the definition of the Fourier transform of the potential ϕ, we can calculate the exact gradient of the potential

$$\hat{\phi}(\mathbf{x}) = \int_{-\infty}^{\infty} \frac{d\mathbf{k}}{2\pi}\phi(\mathbf{k})e^{i\mathbf{k}\mathbf{x}}, \tag{5.18}$$

$$\hat{E}(k) = -i\mathbf{k}\hat{\phi}(\mathbf{k}). \tag{5.19}$$

So $\hat{D}(\mathbf{k}) = i\mathbf{k}$ is also known as $i\mathbf{k}$ differentiation and can be evaluated during potential solver. The electric field (first derivative of potential) requires $\hat{\phi}(\mathbf{k})$ to be multiplied by $-i\mathbf{k}$ before taking inverse FFT. Similarly, the second derivative (as would appear in surface tension evaluation later) can be obtained by multiplying $\hat{\phi}(\mathbf{k})$ with $(i\mathbf{k})^2 = -\mathbf{k}^2$. Inverse FFT would give the required first or second derivative.

5.3.4 Force Interpolation

The mesh based force is interpolated to the atomic position using the weighted function $W(\mathbf{x})$ as described in section 5.3.1. TSC assignment is used for weighting function. The interpolated force is expressed as

$$\mathbf{F}(\mathbf{x}) = \frac{q}{V_c}\int W(\mathbf{x}-\mathbf{x}^{'})E^{\dagger}(\mathbf{x}^{'})d\mathbf{x}^{'}. \tag{5.20}$$

The interpolated force gives the long range contribution of the total force. If we include the above four steps, we can express the mesh calculated force on an unit charge at position $\mathbf{x_2}$ due to a negative charge at $\mathbf{x_1}$

$$\mathbf{F}(\mathbf{x_2};\mathbf{x_1}) = \frac{1}{V_b}\sum \frac{\hat{W}}{V_c}\hat{\mathbf{D}}\hat{G}\sum_n \frac{W(\hat{\mathbf{k_n}})}{V_c}\exp(i(\mathbf{k}-\mathbf{k_n})\cdot\mathbf{x_1}). \tag{5.21}$$

5.4 Optimized Influence Function

P^{3}M method is optimized by minimizing the total error in the above four steps. If the reference force is $\mathbf{R}(\mathbf{x})$, then the total square of the error Q is minimized.

$$Q = \frac{1}{V_c}d\mathbf{x_1}\int_{V_c} d\mathbf{x_1}\int_V d\mathbf{x}|\mathbf{F}(\mathbf{x};\mathbf{x_1}) - \mathbf{R}(\mathbf{x})|^2. \tag{5.22}$$

Hockney suggested to express Q in Fourier components and influence function $\hat{G}(\mathbf{k})$ and applied the power theorem to express it as

$$Q = \frac{1}{V_c}\int_{V_c} d\mathbf{x_1}\frac{1}{V_b}\sum\left[|\hat{\mathbf{F}}(\mathbf{k};\mathbf{x_1})|^2 - 2\hat{\mathbf{R}}^*(\mathbf{k})\cdot\hat{\mathbf{F}}(\mathbf{k};\mathbf{x_1}) + |\hat{\mathbf{R}}(\mathbf{k})|^2\right], \tag{5.23}$$

where

$$\mathbf{F}(\mathbf{k};\mathbf{x_1}) = \frac{\hat{W}}{V_c}\hat{\mathbf{D}}\hat{G}\sum_n \frac{W(\hat{\mathbf{k_n}})}{V_c}\exp(i(\mathbf{k}-\mathbf{k_n})\cdot\mathbf{x_1}). \tag{5.24}$$

The optimized influence function is obtained by

$$\frac{\partial Q}{\partial G} = 0. \tag{5.25}$$

It gives

$$\hat{G}(\mathbf{k}) = \frac{\hat{\mathbf{D}}(\mathbf{k})\cdot\sum_n U^2(\mathbf{k_n})\hat{\mathbf{R}}(\mathbf{k_n})}{|\hat{\mathbf{D}}(\mathbf{k})|^2\left[\sum_n \hat{U}^2(\mathbf{k_n})\right]^2}, \tag{5.26}$$

where $\hat{U} = \hat{W}/V_c$. For $\hat{U}(\mathbf{k}) = \hat{U}(k_1, k_2, k_3)$ is

$$\hat{U}(\mathbf{k}) = \left(\frac{\sin \frac{k_1 H}{2}}{\frac{k_1 H}{2}} \frac{\sin \frac{k_2 H}{2}}{\frac{k_2 H}{2}} \frac{\sin \frac{k_3 H}{2}}{\frac{k_3 H}{2}} \right)^{p+1}, \tag{5.27}$$

where p is the order of the assignment function. In our case, we have $p = 2$ for TSC function. $\sum \hat{U}^2$ can be simplified to give

$$\sum_n \hat{U}^2 = \prod_{i=1}^{3} \left(1 - \sin^2 \frac{k_i H}{2} + \frac{2}{15} \sin^4 \frac{k_i H}{2} \right). \tag{5.28}$$

D is the differential operator and its Fourier transform is given as $\hat{D}(k_1, k_2, k_3) = ik_1, ik_2, ik_3$ where i is the imaginary quantity. $\hat{R}$ is the reference interparticle force for the long range part in Fourier space and is discussed in the next section. $\hat{G}$ is evaluated once in the beginning of the simulation as it does not depend on the instantaneous charge density of the atoms. As we will see later that R is different for force and surface tension, it requires two separate influence functions for force and surface tension evaluation.

5.5 Force Split for $1/r^p$ Type Term (Essmann *et al.* [1])

Lattice summation for Coulomb type force $(1/r)$ is very common and known as the Ewald sum. Similar expressions can be derived for any $1/r^p$ type force which can be used in P^{3}M method to evaluate the force without any truncation. Let Λ denote the set of all lattice vectors $\mathbf{n} = n_1\mathbf{a_1} + n_2\mathbf{a_2} + n_3\mathbf{a_3}$. Here we assume a closed, bounded region P in R^3, centered on the origin. For positive integers K, let $P_K(\Lambda)$ denote the set of lattice vectors $\mathbf{n}$ such that $\mathbf{n}/K \in P$. Given N points $\mathbf{r_1}, \ldots, \mathbf{r_N}$ in the unit cell U, and real constants C_{ij}, $1 \le i, j \le N$, we consider sums of the form

$$E_P(\mathbf{r_1},\ldots,\mathbf{r_N}) = \lim_{K\to\infty}\frac{1}{2}\sum_{n\epsilon P}{}' \sum_i\sum_j \frac{C_{ij}}{|\mathbf{r_i}-\mathbf{r_j}+\mathbf{n}|^p}. \tag{5.29}$$

where the prime on the outer sum indicates $i = j$ and $\mathbf{n} = 0$ are omitted.

Some of the identities for $1/|r|^P$ are shown here. It is based on the well known identity

$$\Gamma(z) = \int_0^\infty t^{z-1}\exp(-t)\,dt = \lambda^z\int_0^\infty t^{z-1}\exp(-\lambda t)\,dt, \tag{5.30}$$

where $\Gamma(z)$ is the Euler Function. We note also that

$$\exp(-a^2w^2) = \frac{\sqrt{\pi}}{a}\int_0^\infty \exp(-\pi^2u^2/a^2)\exp(-2\pi iuw)\,du \tag{5.31}$$

is the Fourier integral expression of the Gaussian. In Eq. (5.30), given a three dimensional vector $\mathbf{r}$, we substitute $\lambda = |\mathbf{r}|^2 = r^2$ and $z = p/2$. For an arbitrary positive number β, we have

$$\frac{\Gamma(p/2)}{r^p} = \int_0^{\beta^2} t^{p/2-1}\exp(-r^2t)\,dt + \int_{\beta^2}^\infty t^{p/2-1}\exp(-r^2t)\,dt = I_p + II_p. \tag{5.32}$$

For II_p, we substitute $r^2t = s;\ dt = 2s\,ds/r^2$ and we have

$$II_p = \int_{\beta r}^\infty \left(\frac{s}{r}\right)^{p-2}\exp(-s^2)\frac{2s}{r^2}ds = \frac{2}{r^p}\int_{\beta r}^\infty s^{p-1}\exp(-s^2)ds. \tag{5.33}$$

For I_p, we write $r^2 = x^2 + y^2 + z^2$ and $a^2 = t$ and apply Eq. (5.31) in all three dimensions, substituting w with x, y and z respectively. Now consider the reciprocal unit cell U^* made up of points $\mathbf{u}$ in R^3 such that $-1/2 \le \mathbf{a}\cdot\mathbf{u} \le 1/2$.

$$\begin{aligned} I_p &= 2\int_0^\beta a^{p-1}\left\{\frac{\sqrt{\pi}}{a}\int_0^\infty \exp\left(-\frac{\pi^2 u_1^2}{a^2}\right)\exp(-2\pi i u_1 x)du_1\right\} \\ &\quad \left\{\frac{\sqrt{\pi}}{a}\int_0^\infty \exp\left(-\frac{\pi^2 u_2^2}{a^2}\right)\ \exp(-2\pi i u_2 y)du_2\right\} \\ &\quad \left\{\frac{\sqrt{\pi}}{a}\int_0^\infty \exp\left(-\frac{\pi^2 u_3^2}{a^2}\right)\ \exp(-2\pi i u_3 z)du_3\right\} da. \end{aligned} \tag{5.34}$$

Changing the order of integration to integrate over a first and substituting a with s, where $\pi u/a = s$, we get

$$I_p = 2\pi^{3/2}\int_{R^3}\exp(-2\pi i\mathbf{u}\cdot\mathbf{r})\int_0^\beta \exp\left(-\frac{\pi^2|\mathbf{u}|^2}{a^2}\right)a^{p-4}da\ d^3\mathbf{u} \tag{5.35}$$

$$= 2\pi^{3/2}\int_{R^3}\exp(-2\pi i\mathbf{u}\cdot\mathbf{r})\int_{\frac{\pi|\mathbf{u}|}{\beta}}^\infty \exp(-s^2)\left(\frac{\pi|\mathbf{u}|}{s}\right)^{p-4}\frac{\pi|\mathbf{u}|}{s^2}\ ds\ d^3\mathbf{u} \tag{5.36}$$

$$= \pi^{3/2}\beta^{p-3}\int_{R^3}\exp(-2\pi i\mathbf{u}\cdot\mathbf{r})2\left(\frac{\pi|\mathbf{u}|}{\beta}\right)^{p-3}\int_{\frac{\pi|\mathbf{u}|}{\beta}}^\infty s^{2-p}\exp(-s^2)\ ds\ d^3\mathbf{u}. \tag{5.37}$$

As R^3 can be decomposed as the union of the point sets $U^*+\mathbf{m}$ over all reciprocal vectors $\mathbf{m}$, we can express $1/r^p$ from Eq. (5.32), Eq. (5.37) and Eq. (5.33) as,

$$\frac{1}{r^p} = \pi^{3/2}\beta^{p-3}\sum_m\int_{U^*} f_p(\pi|\mathbf{v}+\mathbf{m}|/\beta)\exp(-2\pi i(\mathbf{v}+\mathbf{m})\cdot\mathbf{r})d^3\mathbf{v} + \frac{g_p(\beta r)}{r^p}, \tag{5.38}$$

where for positive numbers x, $f_p(x)$ and $g_p(x)$ are defined by

$$f_p(x) = \frac{2x^{p-3}}{\Gamma(p/2)}\int_x^\infty s^{2-p}\exp(-s^2)\ ds, \tag{5.39}$$

and

$$g_p(x) = \frac{2}{\Gamma(p/2)}\int_x^\infty s^{p-1}\exp(-s^2)\ ds. \tag{5.40}$$

Table 5.1: Function f_p and g_p in Eq. (5.39) and Eq. (5.40)

p	$f_p(x)$	$g_p(x)$
1	$\frac{\exp(-x^2)}{\sqrt{\pi}x^2)}$	$\mathrm{erfc}(x)$
2	$\frac{\sqrt{\pi}}{x}\mathrm{erfc}(x)$	$\Gamma[0,x^2]$
3	$\frac{2\Gamma[0,x^2]}{\sqrt{\pi}}$	$\mathrm{erfc}(x)+\frac{2x\exp(-x^2)}{\sqrt{\pi}}$
4	$2[\exp(-x^2)-x\sqrt{\pi}\mathrm{erfc}(x)]$	$\exp(-x^2)(1+x^2)$
5	$\frac{4}{3\sqrt{\pi}}[\exp(-x^2)-x^2\Gamma[0,x^2]]$	$\exp(-x^2)(6x+4x^2)+3\sqrt{\pi}\mathrm{erfc}(x)$
6	$\frac{1}{3}[(1-2x^2)\exp(-x^2)+2x^3\sqrt{\pi}\mathrm{erfc}(x)]$	$\exp(-x^2)(1+x^2+x^4/2)$
7	$\frac{16x^4}{15\sqrt{\pi}}[(1-x^2)\exp(-x^2)+x^4\Gamma[0,x^2]]$	$\mathrm{erfc}(x)+\frac{\exp(-x^2)(30x+20x^3+8x^5)}{15\sqrt{\pi}}$
8	$\frac{1}{45}[\exp(-x^2)(3-2x^2+4x^4)$ $-4\sqrt{\pi}x^5\mathrm{erfc}(x)]$	$\exp(-x^2)(1+x^2+x^4/2+x^6/6)$

The first term in Eq. (5.38) is the long range contribution $\mathbf{R}(\mathbf{x})$ and the second part is the short range part $\mathbf{F_{sr}}(\mathbf{x})$. If we compare $\mathbf{R}(\mathbf{x})$ with the first part, we notice that it is written as inverse fourier transform of $R(k)$. As $\mathbf{m} \in U^*$ (reciprocal space), it has dimension of $1/L$ and $f_p(\pi|\mathbf{v}+\mathbf{m}|/\beta)$ becomes $f_p(|\mathbf{k}|/2\beta)$ and $\hat{R}(\mathbf{k})$ is constructed for $\hat{G}(\mathbf{k})$.

In particular, for the cases $p = 1$ to 8, we have tabulated $f_p(x)$ and $g_p(x)$ in Table 5.1.

Here $\Gamma(a,x)$ is the incomplete gamma function defined by

$$\Gamma(a,z) = \int_z^\infty t^{a-1}\exp(-t)dt. \tag{5.41}$$

Once f_p and g_p are obtained, they can be implemented in a P^{3}M method as a simple simple generalization of the standard 1/r potential. Function f_p is used in evaluating the long range term and it is included in the evaluation of R in the optimal influence function $\hat{G}$. Function g_p is used in the short range part evaluation in section 5.2. For first or second derivative of potential term, $g_p(\beta r)/r^p$ is analytically differentiated and evaluated within short range cutoff distance using a short range table for faster evaluation.

5.6 Evaluation

In this section, details about the force evaluation and surface tension evaluation is described. Both evaluations use the P^{3}M algorithm for efficient evaluation without any truncation.

5.6.1 Force Evaluation

Lennard Jones atoms interact with each other through the well known (12,6) potential given by

$$\phi_{LJ}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]. \tag{5.42}$$

It has a $1/r^{12}$ repulsive term due to electron cloud at small distance and $1/r^6$ attractive dispersive term. The term $1/r^{12}$ decays faster and can be evaluated with a finite cutoff radius. Even though the dispersive term decays as $1/r^6$, its truncation causes significant error in the system properties. It leads us to use the P^{3}M method to evaluate the dispersive term. This term is split into two parts as shown in Eq. (5.1). For force evaluation, the first derivative of the dispersive term is required and the long range force evaluation follows the same steps as described in section 5.3. Short range dispersive part is the derivative of $g_6(\beta r)/r^6$.

Repulsive $1/r^{12}$ term is included in the short range part.

$$F_{sr} = -\frac{d}{dr}\left(4\epsilon\left(\frac{\sigma}{r}\right)^{12} - 4\epsilon\sigma^6\frac{g_6(\beta r)}{r^6}\right) \tag{5.43}$$

$$= 4\epsilon\left(\frac{\sigma}{r}\right)^{12}\frac{12}{r} - 4\epsilon\sigma^6\frac{g_6(\beta r)6r^5 - r^6 g_6^{'}(\beta r)}{r^{12}} \tag{5.44}$$

$$= 4\epsilon\left(\frac{\sigma}{r}\right)^{12}\frac{12}{r} - 4\epsilon\sigma^6\frac{6g_6(\beta r) - r g_6^{'}(\beta r)}{r^7}, \tag{5.45}$$

where $g_6(x)$ from Table 5.1 is $\exp(-x^2)(1 + x^2 + x^4/2)$ and $g_p^{'}$ is its derivative w.r.t r.

5.6.2 Surface Tension Evaluation

In chapter 2, we derived the expression for the interfacial tension as

$$\gamma_{lv} = \left\langle \sum_{i>j}\sum_{j} \frac{x_{ij}^2 + y_{ij}^2 - 2z_{ij}^2}{2A\, r_{ij}}\phi_{ij}^{'} \right\rangle. \tag{5.46}$$

where ϕ is the Lennard Jones (12,6) potential given in Eq. (5.42). From equation Eq. (5.46) and Eq. (5.42), we get the surface tension term as

$$\gamma_{lv} = \left\langle \sum_{i>j}\sum_{j} \frac{x_{ij}^2 + y_{ij}^2 - 2z_{ij}^2}{2A\, r_{ij}} \left\{\frac{-48\epsilon}{r_{ij}}\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - 0.5\left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\right\} \right\rangle. \tag{5.47}$$

Dropping the subscripts and using the non-dimensional form, we have

$$\gamma_{lv} = \left\langle \sum \left\{ \frac{12(x^2 + y^2 - 2z^2)}{Ar^8} - \frac{24(x^2 + y^2 - 2z^2)}{Ar^{14}} \right\} \right\rangle. \tag{5.48}$$

In the above equation, short range terms like x^2/r^{14} decay much faster than the dispersive term x^2/r^8 and can be evaluated using a finite cutoff radius of $r_c = 3.0\sigma$. However the x^2/r^8 decays slowly and needs to be evaluated without any cutoff radius/truncation. To evaluate the dispersive term, we need to evaluate x^2/r^8, y^2/r^8 and z^2/r^8 individually. For the first term, we will express x^2/r^8 in terms of the derivative of $1/r^p$. Taking the partial derivative of $r(=\sqrt{x^2+y^2+z^2})$ w.r.t x, we get

$$\frac{\partial r}{\partial x} = \frac{x}{\sqrt{x^2+y^2+z^2}} = \frac{x}{r} \tag{5.49}$$

Similarly,

$$\frac{\partial r}{\partial y} = \frac{y}{r} \text{ and } \frac{\partial r}{\partial z} = \frac{z}{r} \tag{5.50}$$

Taking the partial derivative of $1/r^p$ w.r.t x, we get

$$\frac{\partial}{\partial x}\left(\frac{1}{r^p}\right) = \frac{-p}{r^{p+1}}\frac{\partial r}{\partial x} = \frac{-px}{r^{p+2}} \tag{5.51}$$

$$\frac{\partial^2}{\partial x^2}\left(\frac{1}{r^p}\right) = \frac{\partial}{\partial x}\left(\frac{-px}{r^{p+2}}\right) = \frac{-p}{r^{p+2}} + \frac{x^2 p(p+2)}{r^{p+4}}. \tag{5.52}$$

Using $p=4$, we get

$$\frac{\partial^2}{\partial x^2}\left(\frac{1}{r^4}\right) = \frac{-4}{r^6} + 24\frac{x^2}{r^8} \tag{5.53}$$

$$\text{or, } \frac{x^2}{r^8} = \frac{1}{6r^6} + \frac{1}{24}\frac{\partial^2}{\partial x^2}\left(\frac{1}{r^4}\right). \tag{5.54}$$

Using Eq. (5.54), first part of Eq. (5.48) is given as

$$\frac{12(x^2+y^2-2z^2)}{Ar^8} = \frac{1}{2A}\left[\frac{\partial^2}{\partial x^2}\left(\frac{1}{r^4}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{1}{r^4}\right) - 2\frac{\partial^2}{\partial z^2}\left(\frac{1}{r^4}\right)\right]. \tag{5.55}$$

The dispersive term of surface tension is represented as the partial derivative of $1/r^p$ and Essmann *et al.* [1] formulation for $1/r^p$ is used to evaluate the above term. Short range term and long range terms are evaluated separately. Second part of the Eq. (5.48) (short range repulsive term) is included in the short range part of Eq. (5.55) as it converges much faster than the dispersive part and does not account for significant error on interfacial properties. From Essmann *et al.* [1], the short range part is given as

$$\left(\frac{1}{r^p}\right)_{S.R} = \frac{g_p(\beta r)}{r^p}, \tag{5.56}$$

where $g_p(x)$ is given by Eq. (5.40) and β is a arbitrary constant. For $p = 4$, $g_4(\beta r) = \exp(-\beta^2 r^2)(1 + \beta^2 r^2)$. The short range part of the surface tension is

$$\gamma_{S.R} = \frac{1}{2A}\left[\frac{\partial^2}{\partial x^2}\left(\frac{1}{r^4}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{1}{r^4}\right) - 2\frac{\partial^2}{\partial z^2}\left(\frac{1}{r^4}\right)\right]_{S.R} - \frac{24(x^2 + y^2 - 2z^2)}{Ar^{14}}. \tag{5.57}$$

First we consider the second partial derivative of $1/r^p$ w.r.t x,

$$\frac{1}{2A}\frac{\partial^2}{\partial x^2}\left(\frac{1}{r^4}\right)_{S.R} = \frac{1}{2A}\frac{\partial^2}{\partial x^2}\left(\frac{g_4(\beta r)}{r^4}\right) = \frac{1}{2A}\frac{\partial^2}{\partial x^2}\left(\frac{\exp(-\beta^2 r^2)(1+\beta^2 r^2)}{r^4}\right). \tag{5.58}$$

Carrying out the algebraic simplifications, we obtain

$$\frac{1}{2A}\frac{\partial^2}{\partial x^2}\left(\frac{1}{r^4}\right)_{S.R} = \frac{2\exp(-\beta^2 r^2)}{Ar^8}(\beta^6 r^6 + 3\beta^4 r^4 + 6\beta^2 r^2 + 6)x^2 \tag{5.59}$$

From Eq. (5.59) and Eq. (5.57), we have

$$\gamma_{S.R} = \frac{2\exp(-\beta^2 r^2)}{Ar^8}(\beta^6 r^6 + 3\beta^4 r^4 + 6\beta^2 r^2 + 6)(x^2 + y^2 - 2z^2) - \frac{24(x^2 + y^2 - 2z^2)}{Ar^{14}}. \tag{5.60}$$

As the first term of Eq. (5.60) decays exponentially, we found that $r_c = 3\sigma$ is sufficient for the short range part evaluation.

Evaluation of the long range part on charge-potential mesh cells is carried out in a manner similar to force evaluation. However, the force is the first derivative of the dispersive potential term $1/r^6$ and is a vector, but the surface tension has terms involving the second partial derivative of $1/r^4$ and is a scalar. The three components of the surface tension term are evaluated in a similar way, and algebraically rearranged to get the total interfacial tension term. In the case of the surface tension, we multiply the Fourier transform of the charge density based on the atomic positions with the fourier transform of the influence function followed by multiplication with $-k_x^2$, $-k_y^2$ and $-k_z^2$ to obtain the second partial derivative as required in Eq. (5.55).

5.7 Validation of Untruncated Force Evaluation

We implemented the P^{3}M algorithm to compute Lennard-Jones potentials forces without any truncation and it is validated in this section. The total force is decomposed in two parts: (1) a short range contribution and (2) a long range contribution as shown in Eq. (5.1).

To demonstrate the method, we take two atoms separated by distance r in a box of dimension $12 \times 12 \times 12\sigma^3$. For this case, we used $32 \times 32 \times 32$ charge based mesh points and β is set at 1.0. The force between two atoms interacting through a LJ (12,6) potential Eq. (5.42) is shown in Fig. 5.1. Forces are evaluated at 400 distances between 0 and $6\sigma(=$ Box Length/2.0). The short-range component matches the exact for small r whereas the long-range component matches for large r (The "exact" term was evaluated with three image boxes on each side of the central box i.e. 343 boxes are included). We notice that the sum of the short-range and the long-range components overlaps with the exact term for all r. The exact term is represented as solid line, the total force from P^{3}M method is represented as dashed line, short range part is represented as chained line and long range part is represented as dashed-dotted line. For $r < 1\sigma$, the total force is very large and is not shown for clarity but the smooth long range part is visible. The short range part decays exponentially with r and becomes negligible for $r > 3.0\sigma$. The short range part includes $1/r^{12}$ term along with the short range of $1/r^6$ term which is truncated at $r = 3.0\sigma$.

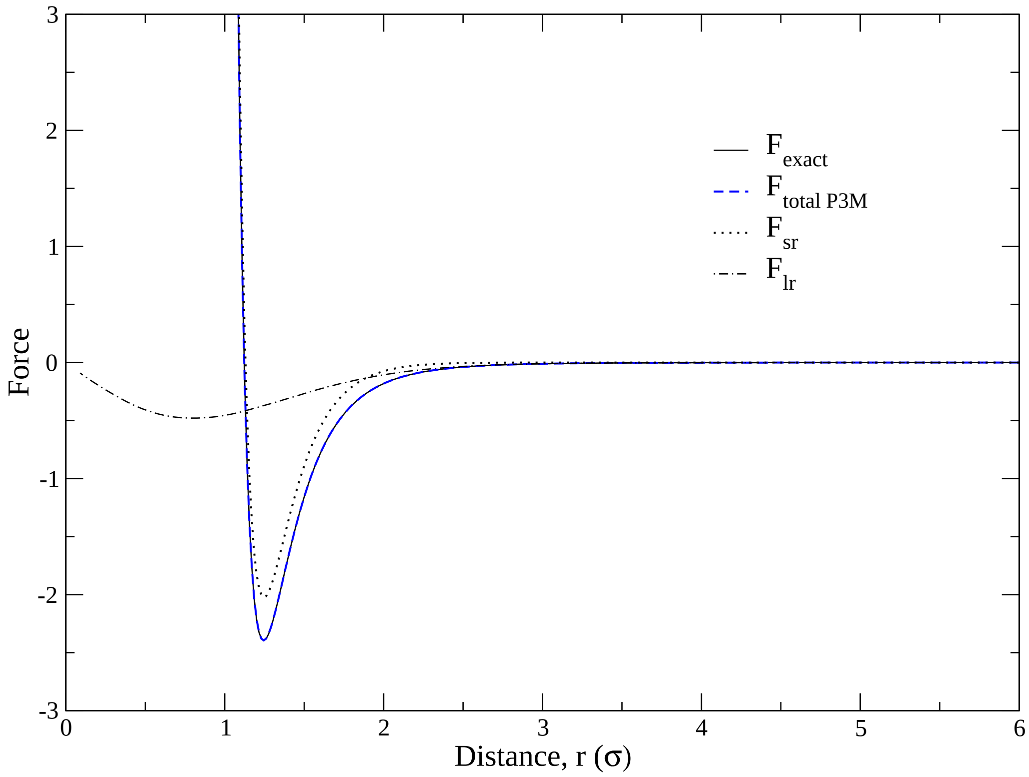
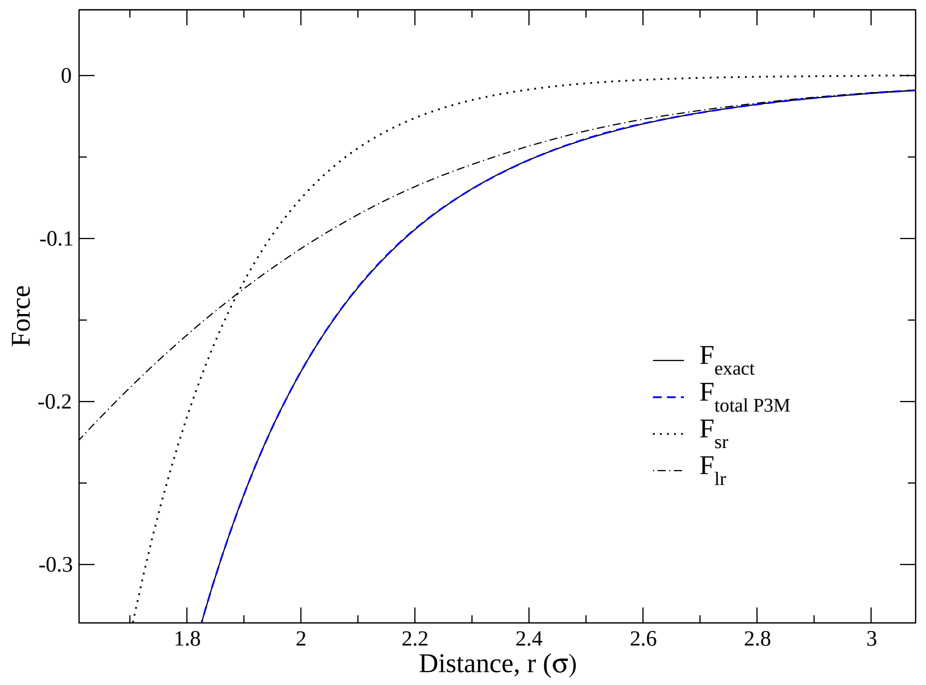
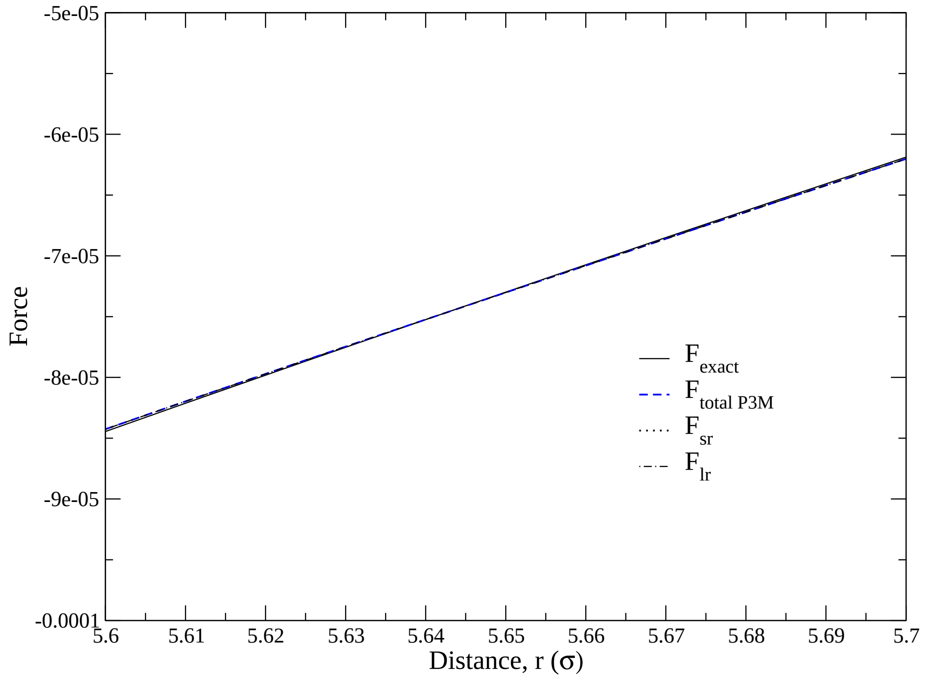


Figure 5.1: Comparison of LJ force using P^3M method with exact calculation

To appreciate the accuracy of this scheme, we magnified the figure at two points in Fig. 5.2. Fig. 5.2(a) shows the magnified view for $r = 1.8\sigma \sim 3.1\sigma$. It also shows the short and long range force terms clearly. The total force overlaps with the exact term. The short range part is negligible compared to the total force whereas the long range term matches with the exact term for $r > 3.0\sigma$. Fig. 5.2(b) magnified view for $r = 5.6\sigma \sim 5.7\sigma$. At this large distance, the force is very small. However, we notice that the match between the total force (or long range force) and the exact force is very good. This gives us high confidence on the force evaluation for molecular dynamics simulation. This method can be equally efficient and accurate for coulomb type forces.



(a)



(b)

Figure 5.2: Magnified view of Fig. 5.1 at (a) $r = 1.8\sigma \sim 3.1\sigma$ and (b) $r = 5.6\sigma \sim 5.7\sigma$

Fig. 5.3 plots the absolute value of the difference between force calculated with P^{3}M method and the exact force as a function of r. We observe that the difference decreases with distance between the atoms. We also observe a local peak at $r = 3\sigma$ corresponding to the short range cutoff distance. At small r, difference is $\mathcal{O}(10^{-2} - 10^{-5})$ but the relative error is very small as the force magnitude is large at small r. Fluctuation in the error is due to evaluation of the long range term in the $\mathbf{k}$ space. As we increase β, the difference increases for $r < r_{cutoff}$. At $r = r_{cutoff}$, the difference gets larger for smaller β because the short range contribution is not negligible for smaller β as it varies as $e^{-\beta^2 r^2}$, but the short range force becomes negligible at $r = 3\sigma$ for $\beta \geq 1.0$. Fig. 5.4 shows the variation of the short range and the long range force for various β. Fig. 5.4(a) shows that the short range force decays faster for larger β and can be neglected after a small r_{cutoff}. This faster decay for larger β is compensated by the long range force as shown in Fig. 5.4(b). It shows that the long range term has higher well for larger β. The sum of these two terms should produce the exact force term within numerical error. To quantify the error, we define *error* as

$$error = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(F_{exact}(r_i) - F_{P^3M}(r_i))^2}. \tag{5.61}$$

As the absolute difference between the exact and P^{3}M calculated force is large at small r, error does not include the term for $r < 1.0\sigma$ as most of the atoms are located at $r > 1.0\sigma$ due to large repulsive force between atoms for $r < 2^{1/6}\sigma (= 1.122\sigma)$. For this case we used $N = 335$ for $1.0\sigma \leq r \leq 6\sigma$. The error is studied as a function of charge-potential mesh points, N_c. Error was evaluated for $N_c = 24^3, 32^3, 48^3$ and 64^3 at $\beta = 0.8, 0.9, 1.0, 1.1$ and 1.2. Error dependence on N_c and β is plotted in Fig. 5.5. We observe that error decreases

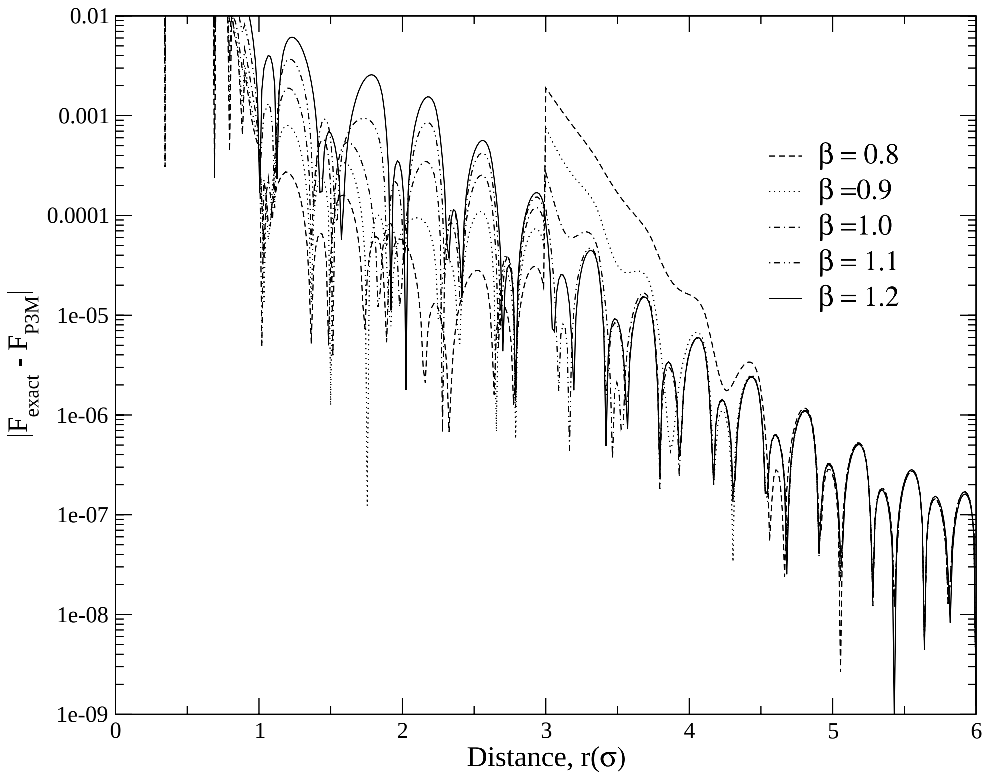
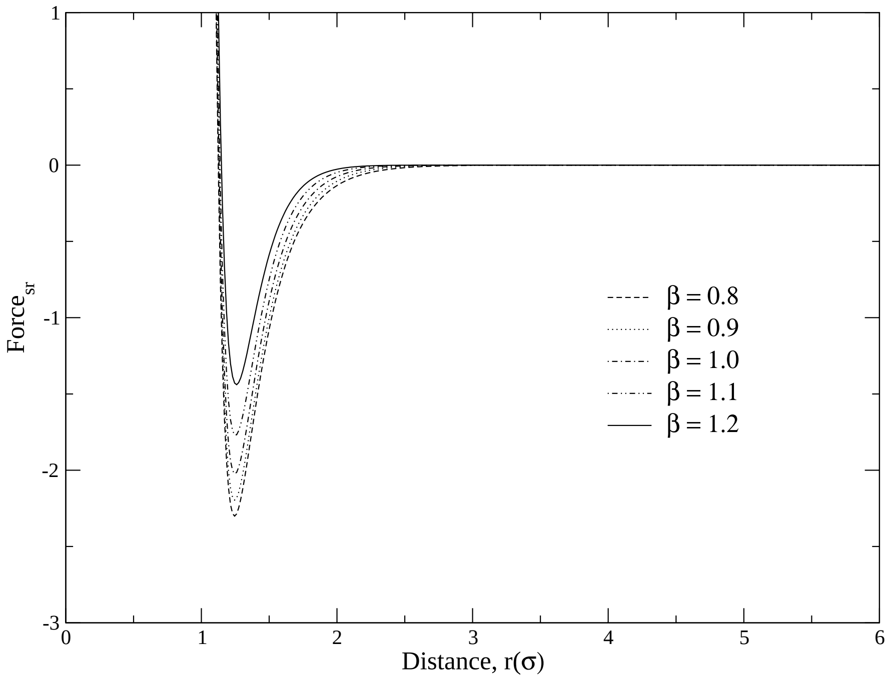
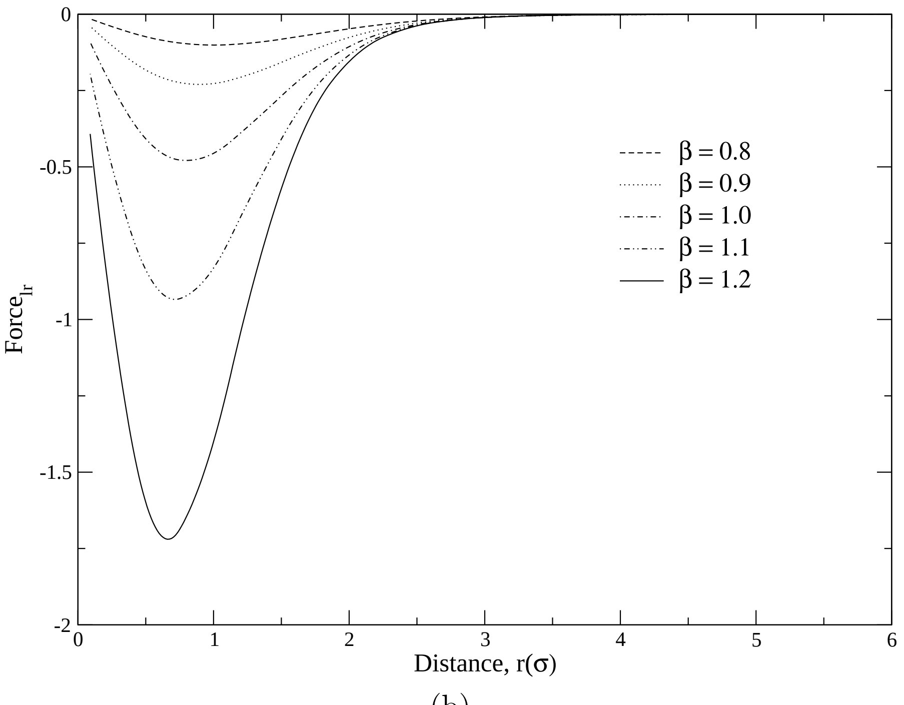


Figure 5.3: Absolute difference of $\text{Force}_{\text{exact}}$ and $\text{Force}_{\text{P}^3\text{M}}$ for mesh points = $32 \times 32 \times 32$, and $r^{sr}_{cutoff} = 3.0\sigma$.

with increasing mesh points. Higher mesh points increase the computational time as FFT is performed on N_c^3 mesh points. It should be noted that error dependence on the mesh points is weaker for smaller β as the error is dominated by the cutoff of the short range force as shown in Fig. 5.3. It requires larger r_{cutoff} radius for smaller β which would increase the computational expense in the short range evaluation. Increasing the mesh points decreases the total error but increases the computational expense and memory usage in the long range force computation using FFTs.



(a)



(b)

Figure 5.4: Variation of (a) the short range force and (b) the long range force with β. $N_c = 32^3$

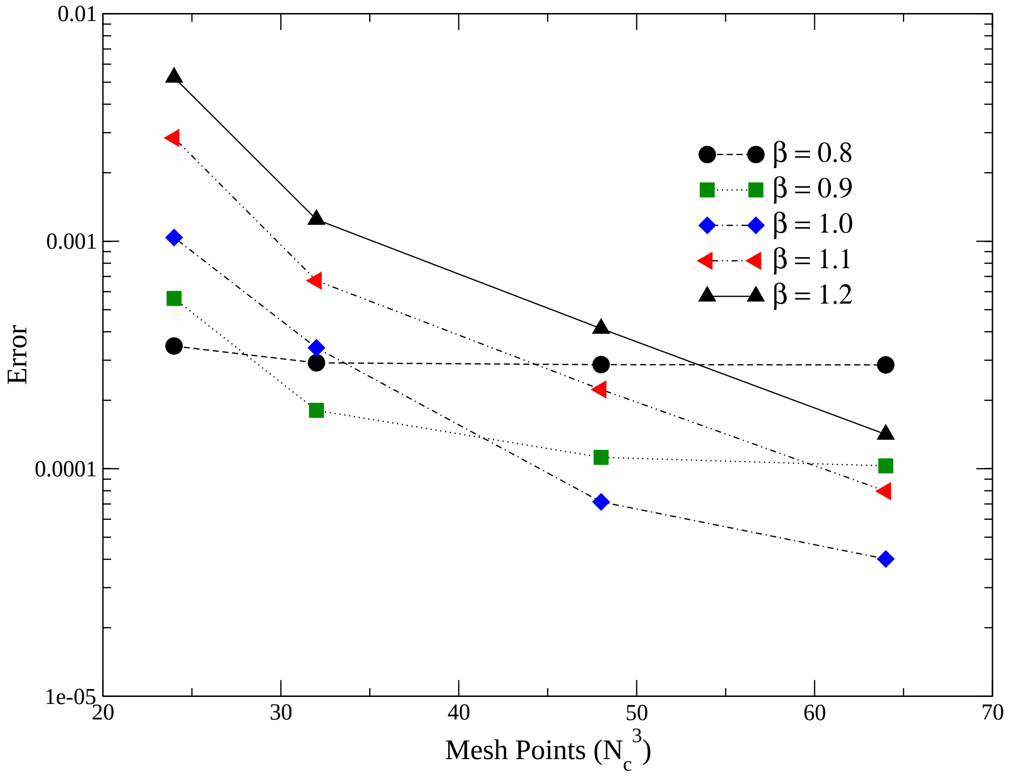


Figure 5.5: Variation of error in force with mesh points N_c and β.

The effect of increasing the cutoff radius from 3.0σ to 3.5σ is shown in Fig. 5.6. Both cases have $N_c = 32^3$ mesh points. We notice that error is the same for larger β (≥ 1.0) whereas error decreases significantly with increasing r_{cutoff} at $\beta = 0.8$. Increasing r_{cutoff} reduces the error due to significant short range cutoff for smaller β. From this observation, mesh points and β can be chosen such that the total error is less, the short range term is negligible at r_{cutoff} and computational expense is reasonable. To achieve the same accuracy as here, mesh points should increase proportional to the domain size such that mesh grid is same.

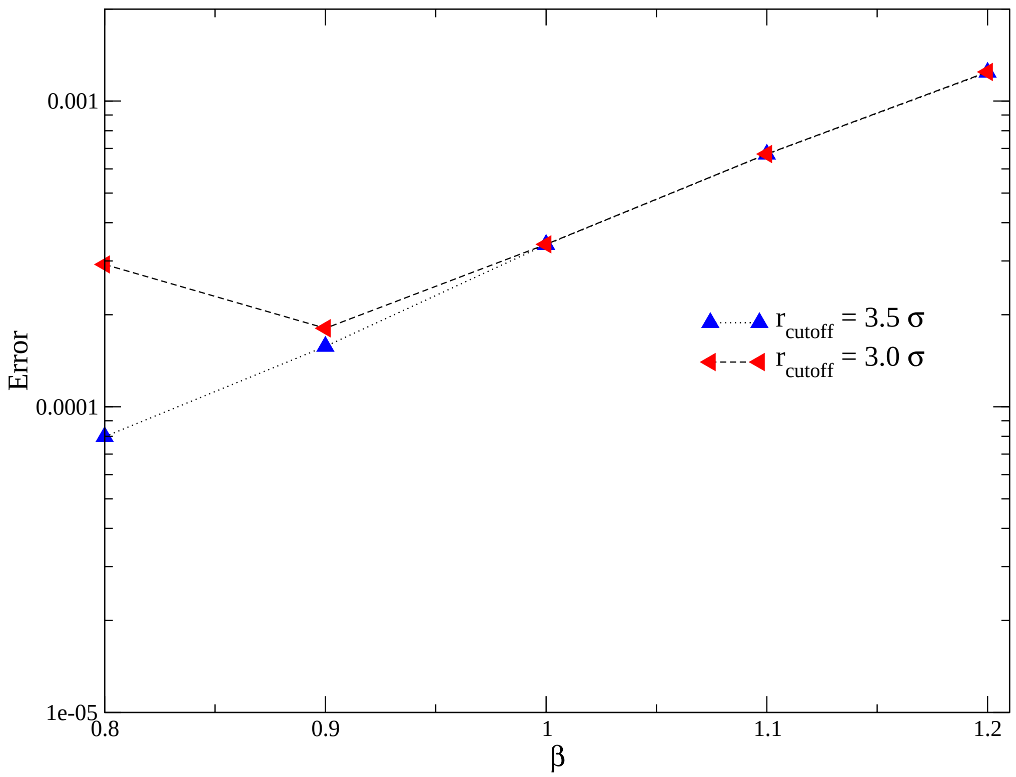


Figure 5.6: Variation of error in force with β for different cutoff radius.

5.8 Validation of Untruncated Surface Tension Evaluation

Surface tension is defined in terms of atomic position in Eq. (5.46). This is applicable for the liquid-vapor interface. If the domain has two interfaces, the statistics gives twice the surface tension. First this term is validated using two LJ atoms separated by distance r to study its accuracy and sensitivity on other parameters, namely β and mesh size. It has been studied in similar ways as force validation performed in the earlier section. For two atoms, we don't have any interface or liquid and vapor phases, but the term can be evaluated using exact direct calculation (using 7^3 mirror boxes) and with P^{3}M method. This comparison is shown in Fig. 5.7 along with the short and long range contribution. Domain size is $12 \times 12 \times 12\sigma^3$ and mesh points are 48^3.

Fig. 5.7 is magnified at two points in Fig. 5.8(a) and (b). Fig. 5.8(a) is exploded view for $r = 1.6\sigma \sim 3.1\sigma$ and Fig. 5.8(b) is exploded view for $r = 2.8\sigma \sim 6.0\sigma$. (a) shows good match between γ calculated by the P^{3}M method and the exact direct calculation. The short range part of γ decays exponentially whereas the long range term matches the exact term at larger r. The short range part is neglected for $r \geq 3.0\sigma$ which is observed in Fig. 5.8(b). The long range part of the surface tension has higher fluctuation in Fig. 5.8(b). This is due to double $i\mathbf{k}$ differentiation of the $1/r^4$ term Eq. (5.55). The relative error appears to be higher and the difference between the γ_{exact} and γ_{P3M} is studied as a function of β and mesh points in Fig. 5.9.

Fig. 5.9(a) shows the effect of β on the difference between γ_{exact} and γ_{P3M} with $N_c = 48^3$. Higher β gives a larger difference whereas lower β gives higher difference near short range cutoff (3.0σ) distance as the short range term is not decayed sufficiently to be neglected for smaller β. This difference is fluctuating with r similar to force evaluation, but the local maximum difference remains

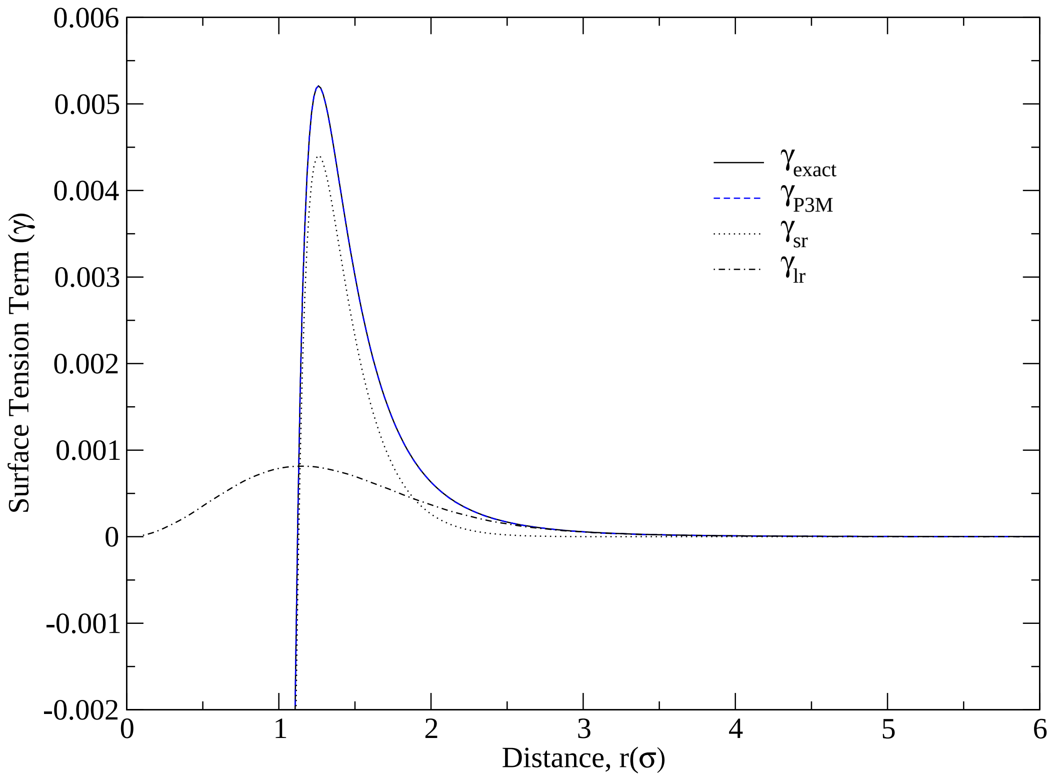


Figure 5.7: Comparison of γ term using P^{3}M method with exact calculation, $N_c = 48^3$, $\beta = 1.0$

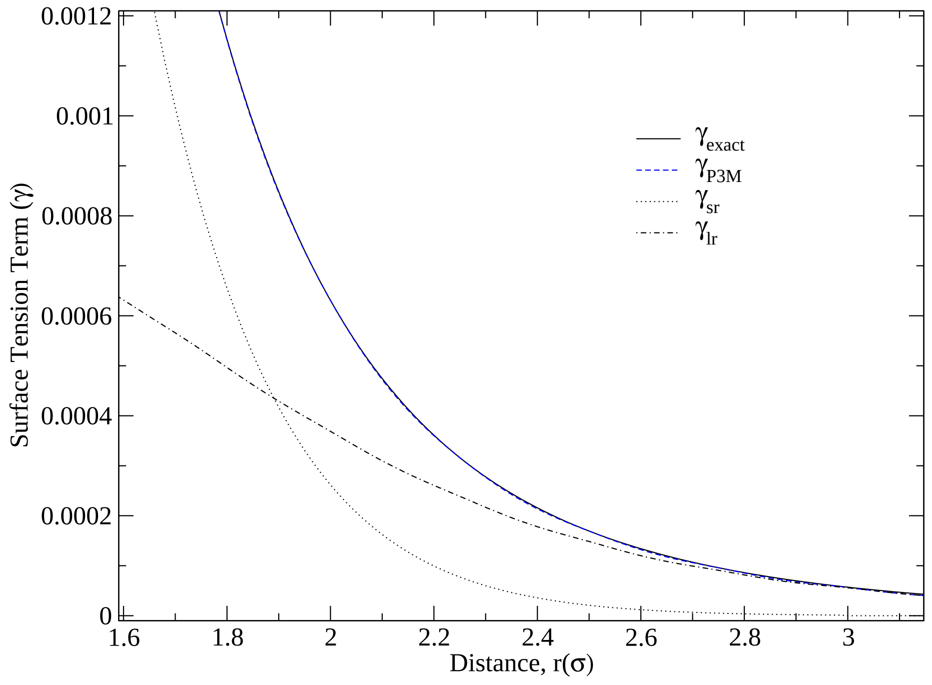
constant unlike the force evaluation (Fig. 5.3) which decreases with increasing r. Fig. 5.9(b) shows this difference as a function of N_c keeping $\beta = 1.0$. It shows that difference between the two terms decreases with increasing r. The width or period of the fluctuation decreases with increasing mesh points and it is same as the mesh size (domain length/N_c). The downside of increasing the mesh points is the extremely high computational expense in the surface tension evaluation.

The effect of β on the short and the long range part of the γ is shown in Fig. 5.10. As β is increased, the short range part decays faster and the peak is lower in Fig. 5.10(a). On the other hand, the peak of the long range part increases with increasing β to compensate the short range part as shown in Fig. 5.10(b).

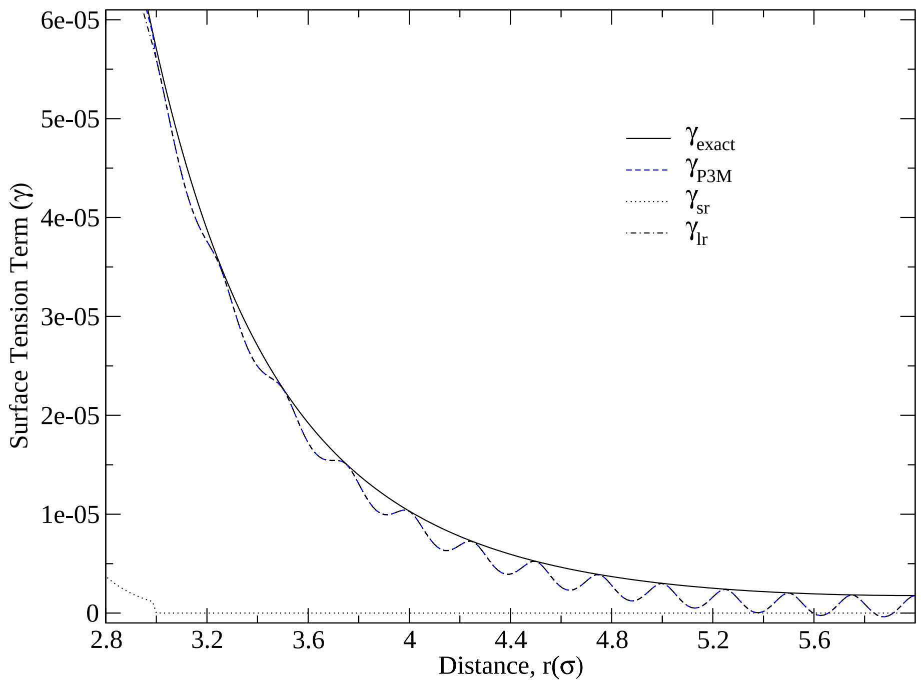
The error in surface tension is quantified in a similar way as the force and it is defined as

$$error = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(\gamma_{exact}(r_i) - \gamma_{P^3M}(r_i))^2}, \tag{5.62}$$

and only $r > 1\sigma$ is considered in the above calculation as the absolute error is reasonably high otherwise and atoms are mostly spaced with $r > 1\sigma$. The variation of error with β and mesh points are plotted in Fig. 5.11. The error is evaluated for the mesh points $N_c = 24^3, 32^3, 48^3, 64^3$ and 96^3. For each case β is varied as 0.8, 0.9, 1.0, 1.1 and 1.2. The cutoff of the short range force is 3.0σ. The magnitude of this error is smaller than the error in the force evaluation and the surface tension term is smaller too. We observe that the error decreases with increasing mesh points. For lower β, this dependence is lesser as error is dominated by cutoff of short range term. Higher β shows strong dependence to the mesh points. Effect of the cutoff radius of the surface tension term on error is shown in Fig. 5.12. We observe that error is higher at cutoff radius of 3.0σ than 3.5σ for $\beta = 0.8$ as error is dominated by cutoff of the short range term. For $\beta \geq 1.0$, error is dominated by the mesh based calculation and cutoff radius has no effect on the error.

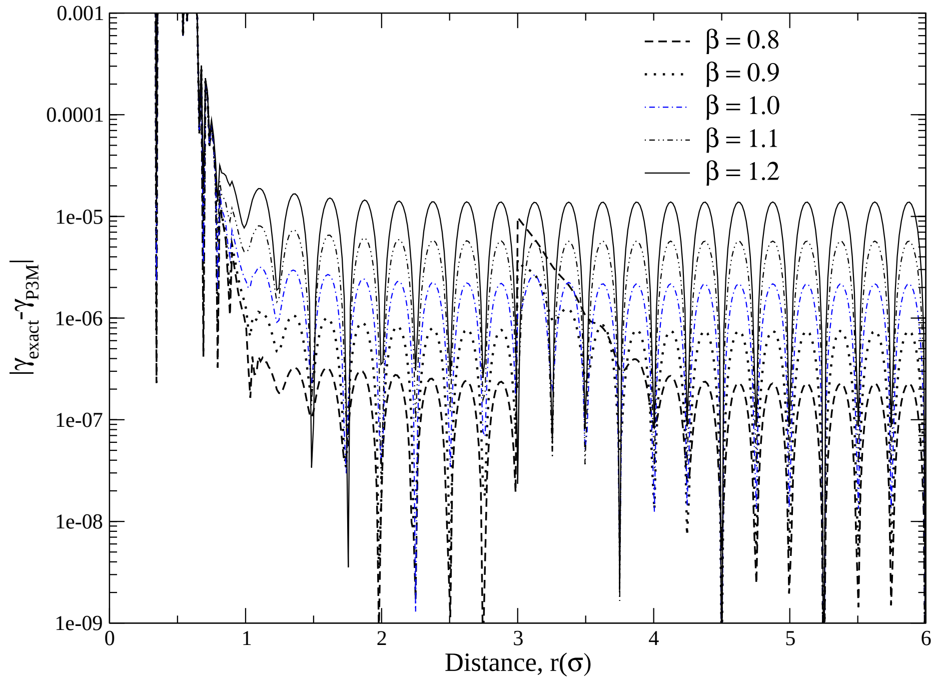


(a)

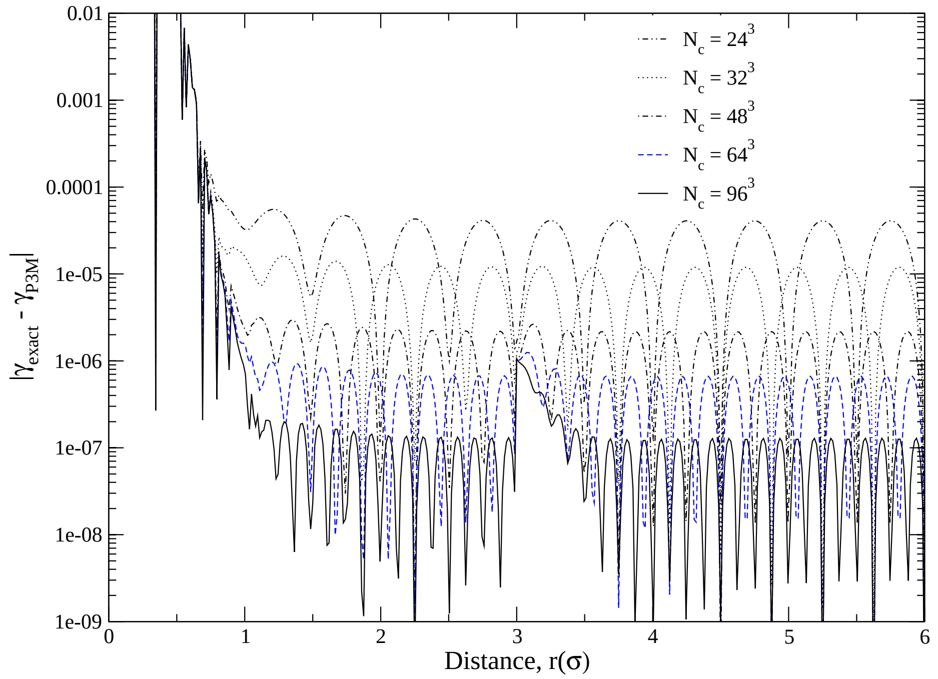


(b)

Figure 5.8: Magnified view of Fig. 5.7 at (a) $r = 1.6\sigma \sim 3.1\sigma$ and (b) $r = 2.8\sigma \sim 6.0\sigma$

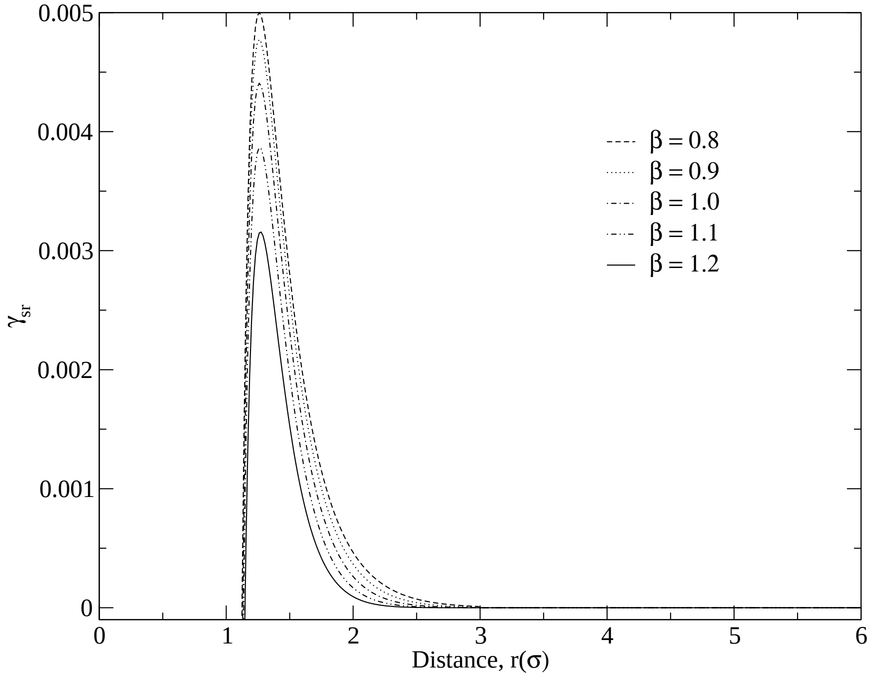


(a)

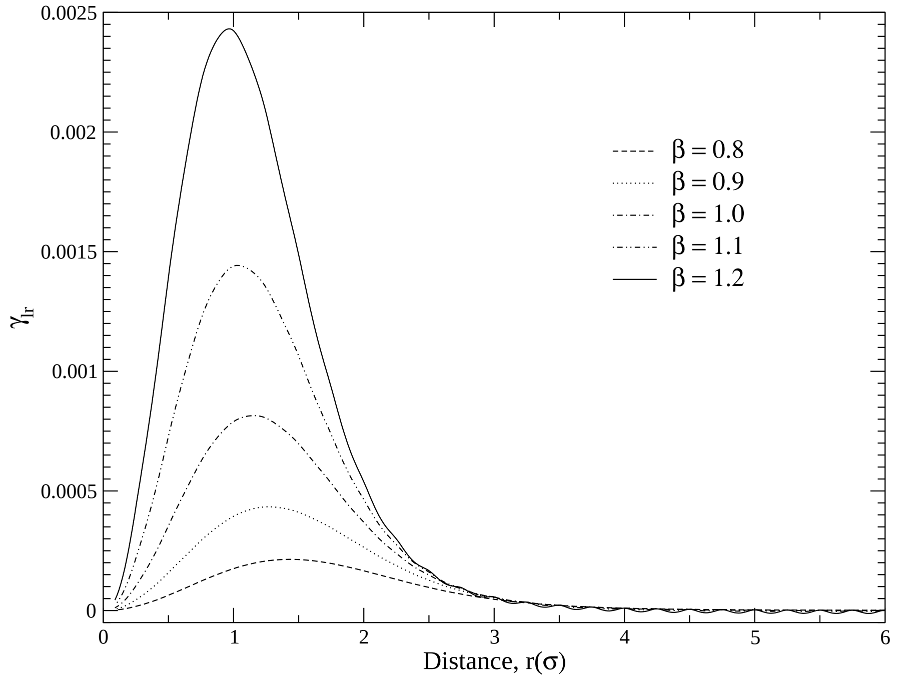


(b)

Figure 5.9: Absolute difference between γ_{exact} and $\gamma_{\text{P}^3\text{M}}$ for $r^{sr}_{cutoff} = 3.0\sigma$. (a) β is verifying, $N_c = 48^3$ and (b) N_c is varying, $\beta = 1.0$



(a)



(b)

Figure 5.10: Variation of (a) γ_{sr} and (b) γ_{lr} with r. $N_c = 48^3$

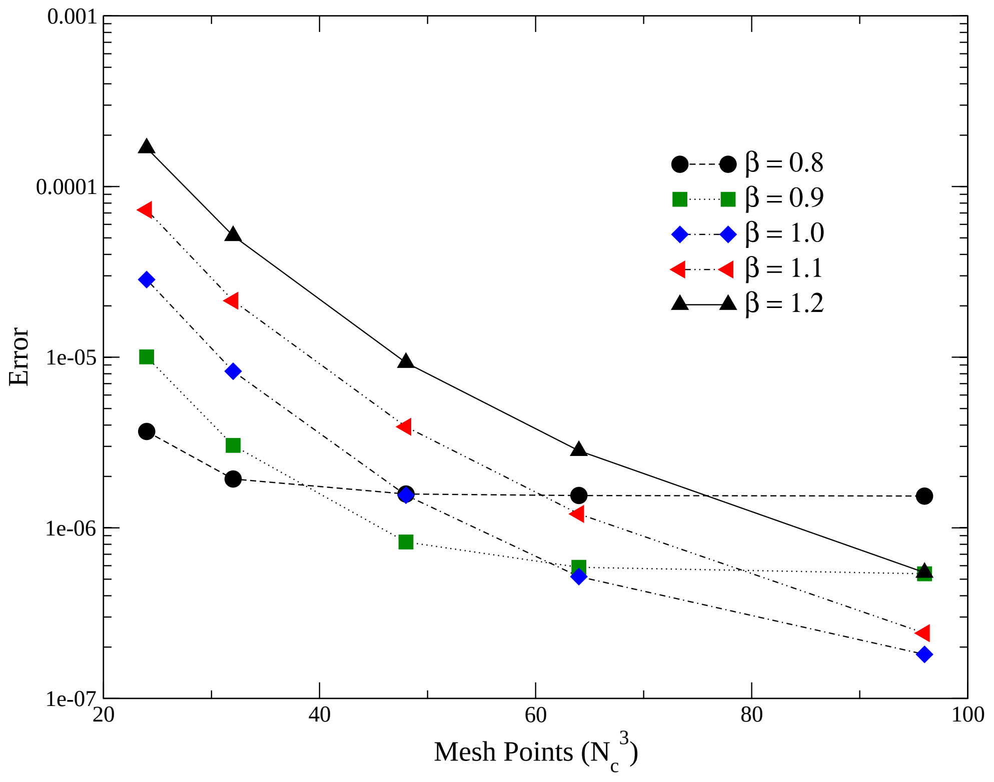


Figure 5.11: Variation of error in surface tension term with mesh points N_c and β.

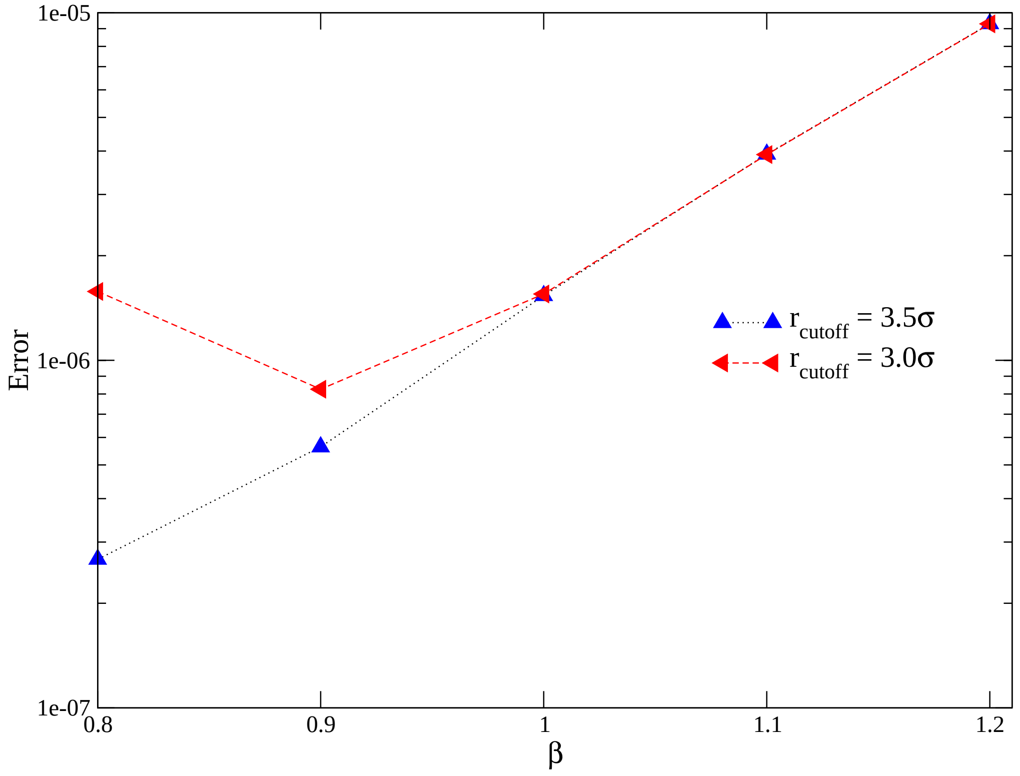


Figure 5.12: Variation of error in γ with β for different cutoff radius.

5.9 Simulation of Liquid Film Using P^{3}M Method

Atomistic simulations were performed in a rectangular domain of size $12 \times 12 \times 24\ \sigma^3$. Dimensional parameters for Argon fluid correspond to $\sigma = 3.405\mathring{A}$ and $\epsilon/k_B = 119.8\ K$. Periodic boundary conditions are applied in all three co-ordinate directions. Lennard-Jones atoms are initially placed in the center of the domain in a liquid slab. One thousand atoms are used in the simulation. The system is equilibrated for 50,000 time steps (0.545 ns) using a Berendson's thermostat to maintain system temperature fixed. The non-dimensional time step of the simulation is 0.005 ($= 1.09 \times 10^{-14}$ sec.). System properties are statistically sampled from the 10^5th time step for the the next 10^6 steps. Computation time for this case is comparable to the simulation using cutoff radius of 4.5σ in the neighbor list algorithm. This procedure was repeated for five temperatures between the triple point ($T^* = 0.7$) and the critical point ($T^* = 1.26$).

We used $32 \times 32 \times 64$ mesh points in the particle-mesh solution for the force evaluation and $48 \times 48 \times 96$ for the surface tension evaluation. Atomic motion was integrated using the *velocity Verlet algorithm.* The center of the mass of the system in the z direction is adjusted every 500 time steps without changing the relative distance of the atoms. Statistics were accumulated in 512 bins parallel to the interface. Density profile of the Lennard-Jones atoms are shown in Fig. 5.13. Liquid and vapor densities are obtained from it and plotted in Fig. 5.14. These values are compared with the density of Argon fluid. We see that the simulations predict liquid and vapor densities well. As temperature increases, liquid density decreases and vapor density increases and the difference vanishes near the critical point.

Interfacial tensions are obtained by integrating the imbalance between normal and tangential pressure. Since there are two interfaces, the value is halved and

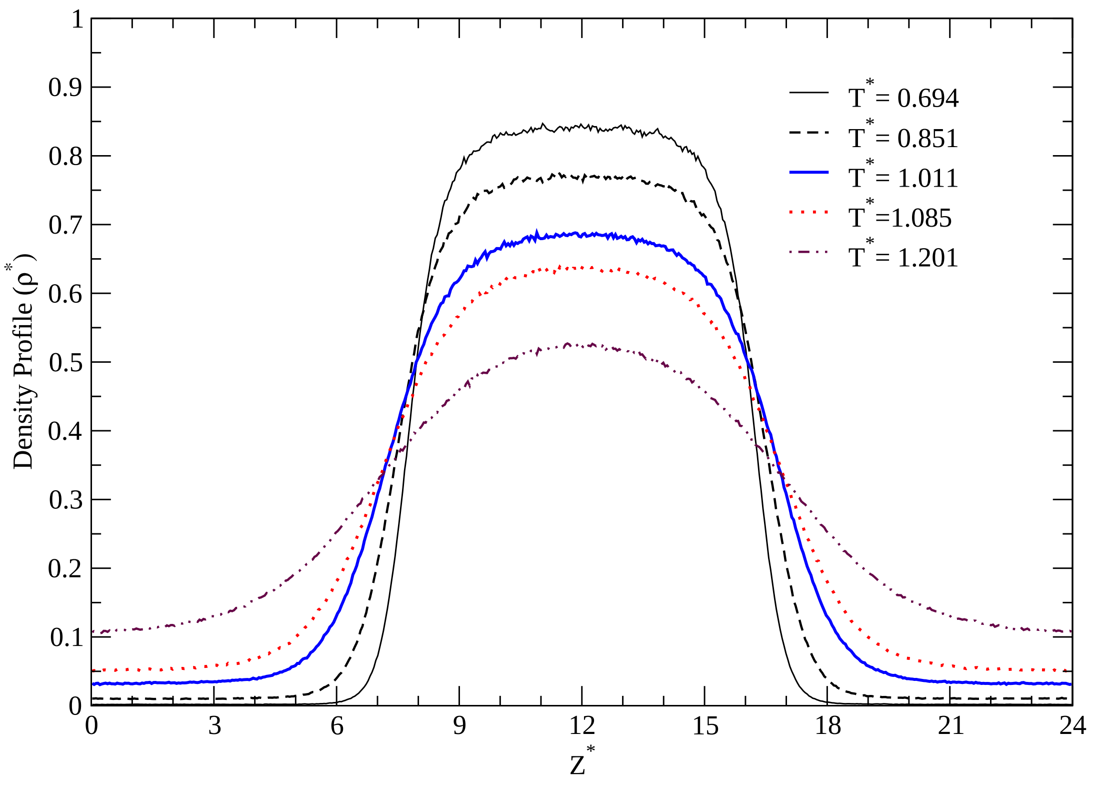


Figure 5.13: Density profiles at various temperatures

then plotted in the Fig. 5.15 where it is compared with the interfacial tension of Argon as well as the results from neighbor list code. Empty circle shows the surface tension result without tail correction due to the finite cutoff and filled circle shows the surface tension with tail correction. It shows that approximately 10 - 15 % tail correction is required for the neighbor list method. All the simulation results match well with the surface tension of the Argon fluid but our P^{3}M code does not require any tail correction for the surface tension or densities.

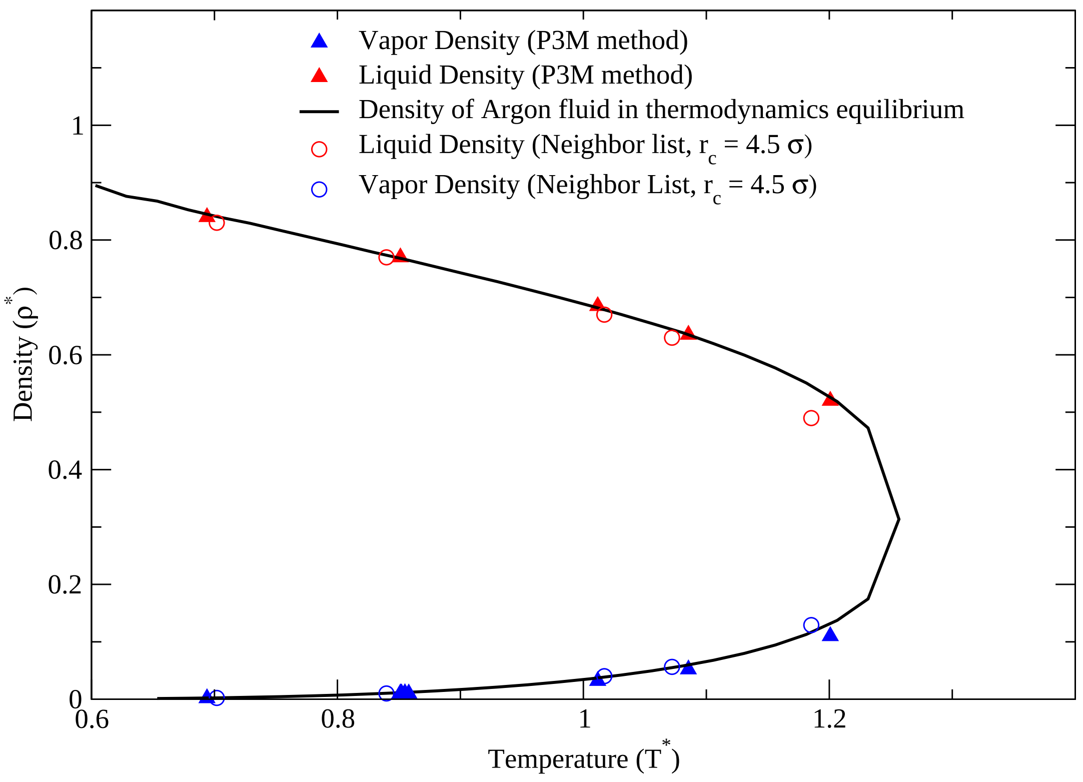


Figure 5.14: Comparing the liquid and the vapor density with the density of Argon fluid in thermodynamic equilibrium

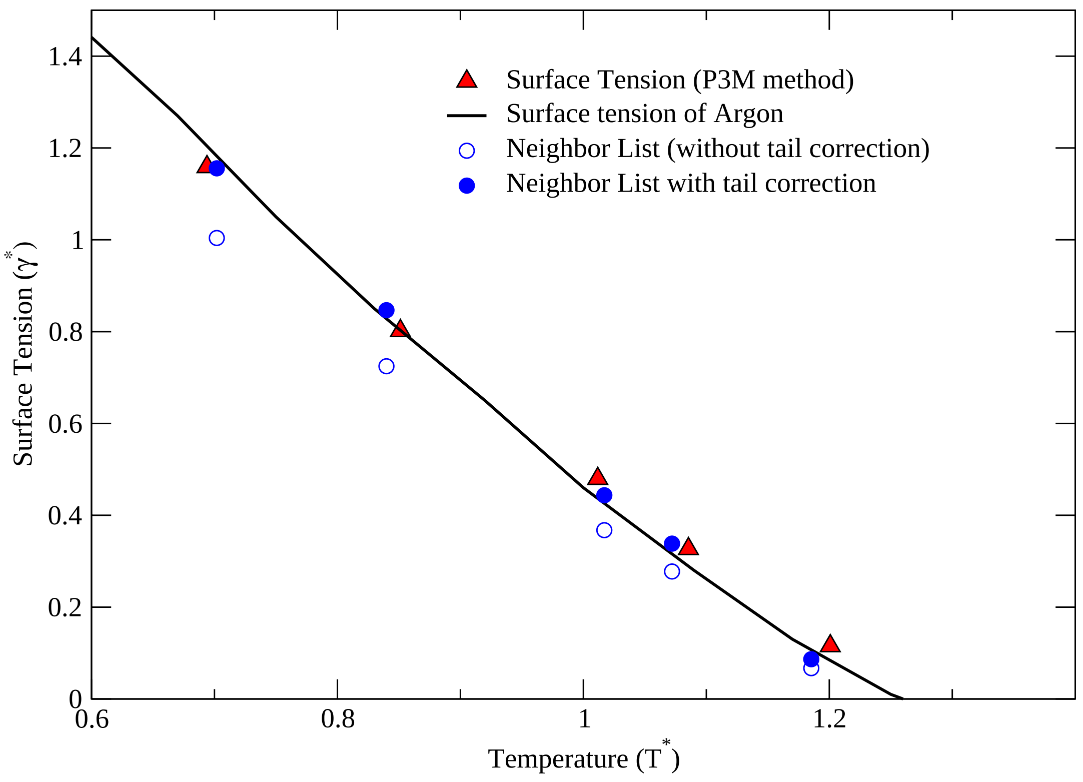


Figure 5.15: Comparison of the surface tension

CHAPTER 6

CONCLUSIONS

In this work, we investigated properties of ultra-thin Lennard-Jones films adjacent to solid surfaces using molecular dynamics simulation. The surface tension is evaluated by integrating the normal and tangential pressure components across the interface. The surface tension dependence on temperature and the film thickness is studied. It has a weak dependence on the film thickness. For film thickness greater than 7σ, interfacial tension match well with the data for bulk Argon fluid. Thinner films has a tendency to break up for lower solid-fluid energy parameters ϵ_{sf} and lower length parameters σ_{sf}. Uniform film of film thickness of 4σ is sustained at high ϵ_{sf} without breaking up and interfacial tension and densities are evaluated.

Liquid droplet is simulated adjacent to a semi-infinite solid surface using molecular dynamics simulations. The contact angle is measured from the density profile. The contact angle is studied as function of system temperature, the solid-fluid interaction energy and length parameters. We conclude that the contact angle depends on the potential well depth of the interaction potential at a given temperature. The Hamaker constant of the fluid-solid combination is evaluated. The contact angle is related to the Hamaker constant, A. It is found that the contact angle can be defined based on $A/\sigma_{sf}^3\rho_l$ at a given temperature.

As interfacial tension and densities are sensitive to the finite cutoff radius during the force and the surface tension evaluation, we implemented a P^{3}M method for evaluating the dispersion force term and the surface tension. It would be an effective tool for molecular simulations of large molecular system. The surface tension of the liquid film is evaluated using the P^{3}M method and compared with the interfacial tension of Argon fluid. The surface tension and densities match well and does not require any tail correction.

6.1 Future Works

Molecular simulations can be used to study the contact angle of the liquid droplet of water (polar molecules) on the solid surface and its dependence on the Hamaker constant. Lennard-Jones fluid and the Platinum surface has positive Hamaker constant, but some solid-fluid combination (like water and fused quartz) has negative Hamaker constant. Contact angle variation with temperature and Hamaker constant could show a different trend and should be explored.

P^{3}M method is successfully implemented for the forces and the surface tension which does not require any correction. Molecular system can be simulated and various system properties can be sampled using similar way. In our case, a 3D periodic conditions are used. However, this can be extended to molecular systems which has 2D periodic condition (like droplet simulation in Chapter 4) by modifying the reference force. Our implementation is used for one type of the LJ atoms. This can be used for dissimilar atoms interacting through LJ potentials as long as $\epsilon_{12} = \sqrt{\epsilon_{11}\epsilon_{22}}$ where 1 and 2 are type of the LJ atoms and ϵ_{ij} is the energy parameter of the LJ potential between atom type i and j. This method should be extended for dissimilar atoms which does not obey the above restriction.

Appendix A: Code for Force and Surface Tension Evaluation using P^{3}M Method

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! This method implement force and surface tension
!! evaluation for a liquid film surrounded by its vapor
!! in thermal equilibrium. Makefile and input data is
!! also included. Some of the variables are used for
!! code testing only.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!
!! This is main.f90
!! Main program section...
!!!!
PROGRAM run
  USE prms
  USE data
  USE nlistmod
  USE p3mlj
  USE p3mgam

  IMPLICIT none

  call init
  call md_start

CONTAINS
  SUBROUTINE md_start
    real :: xx,rr,delta_r2,rc
    real,dimension(3)  :: xr
    real,dimension(Natm,3)  :: Xold,F
    real,dimension(Natm):: q
    integer   :: i,j,icycle,iflag,nrun,istart,ibin
    real,dimension(Nbin):: gamma_bin,gz,sur_ten,density,PressN,
PressT,pn,pt,F_gam,R_gam,F_gam_total, R_gam_total
    real,dimension(Natm):: G,G_p3m,Gsr_p3m,Glr_p3m

   write(*,*) "Starting Molecular Dynamics"
```

```
!!$ Initialising the statistical variables
    gz = 0
    q = 1.0
    sur_ten = 0
    nrun = 0
    PressN = 0
    PressT = 0
    F_gam_total = 0
    R_gam_total = 0
!!$ gz: density, q: charge, nrun: number of sampling, PressN:
normal Pressure
!!$ PressT: tangential Pressure, F_gam_total: short range part
Gamma
!!$ R_gam_total: long range part Gamma
    rc = srcutoff*sigmax
    if (I_time .eq. 1) then
       call restart(istart,nrun,X,V,gz,sur_ten,PressN,PressT)
    else
       istart = 0
    end if
!!$ Store system temperature with time data in "temp.dat"
    open(unit=tmp_unit,file="temp.dat")
    delta_r2 = (listfac - 1)**2*rc*rc
    write(*,*) "(rl - rc)^2", delta_r2

    Xold = X
    iflag = 0
    call LJ_nlistFsr(X)
    call LJ_forcePPPM(X,q,F)
!!$ Main loop of MD code.
    do icycle = istart+1,Nt
       write(*,*) "Time Steps = ",icycle
         do i = 1,Natm
              xr(:) = X(i,:) - Xold(i,:)
              xr(:) = xr(:) - Lb(:)*nint(xr(:)/Lb(:))
              rr = xr(1)*xr(1)+xr(2)*xr(2)+xr(3)*xr(3)
!!$     Checking for the neighbor list.
              if (rr.gt.delta_r2) then
                 iflag = 1
                 call LJ_nlistFsr(X)
                 go to 100
              end if
          enddo
100    continue
          if (iflag.eq.1) then
              Xold = X
              iflag = 0
          end if

!!$  Verlet velocity algorithm for time integration
          do i=1,Natm
             X(i,:) = X(i,:) + V(i,:)*Ts + F(i,:)*Ts*Ts/2.0
             V(i,:) = V(i,:) + F(i,:)*Ts/2.0
             do j = 1,3
                if (X(i,j) .gt. Lb(j)) then
                   X(i,j) = X(i,j) - Lb(j)
                else if (X(i,j) .lt. 0) then
                   X(i,j) = X(i,j) + Lb(j)
                end if
             enddo
          enddo
!!$ Calculating the force on each atom.
          call LJ_forcePPPM(X,q,F)
          do i=1,Natm
             V(i,:) = V(i,:) + F(i,:)*Ts/2.0
          enddo
!!$
       if (icycle.le.Ts1) then
          call adjust
       end if
       if (mod(icycle,50).eq.0) then
          call findtemp(xx)
!!$        write(*,*) "Time-step, Temperature", icycle, xx
          write(tmp_unit,*) icycle, xx
!!$ Adjusting the center of mass for better density profile
          if (mod(icycle,500).eq.0) then
             call adjust_com
          write(*,*) "Time-step, Temperature", icycle, xx
!!$ Adjust center of mass in z direction
          end if
       end if
```

```
!!$ Sampling surface tension, pressure components and density
       if ((mod(icycle,25).eq.0).and.(icycle.gt.Ts2)) then
          nrun = nrun + 1
          call GAM_nlistFsr(X)
          call surface_tension(gamma_bin,pn,pt,F_gam,R_gam,
G_p3m,Gsr_p3m,Glr_p3m)
          sur_ten = sur_ten + gamma_bin
          PressN = PressN + pn
          PressT = PressT + pt
          F_gam_total = F_gam_total + F_gam
          R_gam_total = R_gam_total + R_gam
          call cal_density(density)
          gz = gz + density
!!$ Writing output files
          if  (mod(icycle,res_out).eq.0) then
              call output(icycle)
              call output1(icycle,nrun,gz,gamma_bin,sur_ten,
PressN, PressT,F_gam_total,R_gam_total)
          end if
       endif
    enddo
    close(tmp_unit)
    close(22)

  END SUBROUTINE md_start

  SUBROUTINE findtemp(temp)
    real   :: sum1,temp
    integer  :: i
    sum1 = 0
    do i =1,Natm
       sum1 = sum1 + V(i,1)*V(i,1)+V(i,2)*V(i,2)+V(i,3)*V(i,3)
    enddo
    temp = sum1/(3*Natm)
!!$ Find the system temperature
  END SUBROUTINE findtemp

  SUBROUTINE adjust
    real,dimension(3) :: sumV  ! to fix net
velocity as zero
    integer  :: i
    real :: fs,sum1
!!$ Subroutine adjust the current temperature to the target
!!$ temperature as well as
!!$ Net momentum is adjusted to the required condition (zero
presently).
    sumV(:) = 0
    sum1 = 0
    do i =1,Natm
       sumV(:) = sumV(:) + V(i,:)
    enddo
    do i =1,Natm
       V(i,:) = V(i,:) - sumV(:)/Natm
       sum1 = sum1 + V(i,1)*V(i,1)+V(i,2)*V(i,2)+V(i,3)*V(i,3)
    enddo
    fs = sqrt(3*Ttar*Natm/sum1)

    do i =1,Natm
       V(i,:) = V(i,:)*fs
    enddo

  END SUBROUTINE adjust

  SUBROUTINE init
!!$ Intializing all the variables, read inputdata , Ghat
    integer :: i,j,k

    write(out_unit,*) "READING PARAMETERS"; call readprms
    write(out_unit,*) "INITIALIZING DATA" ; call initdata
    write(out_unit,*) "INITIALIZING FFT FOR LJ"; call LJ_initfft
    write(out_unit,*) "INITIALIZING PPPM FOR LJ";call LJ_initPPPM
    write(out_unit,*) "COMPUTING GHAT";  call LJ_newalpha
    write(out_unit,*) "COMPUTED GHAT FOR FORCE"
!!$ Initialising p3m for surface tension
    write(out_unit,*) "INITIALIZING FFT FOR LJ"; call GAM_initfft
    write(out_unit,*) "INITIALIZING P3M FOR LJ";call GAM_initPPPM
    write(out_unit,*) "COMPUTING GHAT";  call GAM_newalpha
    write(out_unit,*) "COMPUTED GHAT FOR SURFACE TENSION"

  END SUBROUTINE init

  SUBROUTINE output(icycle)
  integer      :: icycle, i1,i10,i100,i1000,itemp,i
  character(15) :: name
!!$  itemp = icycle/res_out
  itemp = icycle/res_out
  i100 = itemp/100
  i1 = itemp - i100*100
  i10 = i1/10
  i1 = i1 - i10*10
  name = 'res'//char(i100+48)//char(i10+48)//char(i1+48)//'.
dat'
  open(unit=atm_unit, file="result/"//name)
!!$  write(atm_unit,*) icycle
  do i = 1,Natm
      write(atm_unit,101) i,X(i,:),V(i,:)
  enddo
 101 format(I9,6E25.15)
  close(atm_unit)
!!$ Write output files for atomic positions and velocities
  END SUBROUTINE output

  SUBROUTINE output1(icycle,nr,gz,gamma_bin,gam,PressN,
PressT,F1,R1)
  integer      :: icycle, i1,i10,i100,i1000,itemp,i,nr
  real, dimension(Nbin) :: gz,gamma_bin,gam,PressN,PressT,
Gtmp,Gtotal,F1,F2,R1,R2
  character(15) :: name
  itemp = icycle/res_out
  i100 = itemp/100
  i1 = itemp - i100*100
  i10 = i1/10
  i1 = i1 - i10*10
  name = 'stat'//char(i100+48)//char(i10+48)//char(i1+48)//'.
dat'
  open(unit=stat_unit, file="result/"//name)
  write(stat_unit,*) icycle, nr
!!$  write(stat_unit,*) Lb(1),Lb(2),Lb(3)
!!$  write(stat_unit,*)
  do i = 1,Nbin
      write(stat_unit,*) i,gz(i),gamma_bin(i),gam(i),PressN(i),
PressT(i),F1(i),R1(i)
  enddo
  close(stat_unit)
!!$ Write output file for statistical data
  END SUBROUTINE output1

  SUBROUTINE surface_tension(gamma_bin,pn,pt,F_gam,R_gam
,gamma,gamF,gamR)
  real,dimension(Natm) :: gamma,q,ppn,ppt,gamF,gamR
  real,dimension(Nbin) :: gamma_bin,pn,pt,F_gam,R_gam
  integer :: ibin,i
  q = 1.0
  call GAM_forcePPPM(X,q,gamma,ppn,ppt,gamF,gamR)
  gamma_bin = 0
  pn = 0
  pt = 0
  F_gam = 0
  R_gam = 0
  do i=1,Natm
      ibin = min(int(X(i,3)*Nbin/Lb(3)+1),Nbin)
      gamma_bin(ibin) = gamma_bin(ibin) + gamma(i)
      pn(ibin) = pn(ibin) + ppn(i)
      pt(ibin) = pt(ibin) + ppt(i)
      F_gam(ibin) = F_gam(ibin) + gamF(i)
      R_gam(ibin) = R_gam(ibin) + gamR(i)
  enddo
!!$ Surface tension, prssure assigned to bins in z direction.
  END SUBROUTINE surface_tension

  SUBROUTINE cal_density(density)
  real,dimension(Nbin) :: density
  integer :: ibin,i
  density = 0
  do i=1,Natm
      ibin = min(int(X(i,3)*Nbin/Lb(3)+1),Nbin)
      density(ibin) = density(ibin) + 1
  enddo
!!$ Desntiy assigned in z direction.
  END SUBROUTINE cal_density

  SUBROUTINE restart(istart,nr,X1,V1,gz1,sur1,pn1,pt1)
  integer :: istart,i,nr,ii
  real,dimension(Natm,3) :: X1,V1
  real,dimension(Nbin) :: gz1,sur1,pn1,pt1
  real :: a,b,c
  open(unit=res_unit,file="res.dat")
  read(res_unit,*) istart
  do i=1,Natm
    read(res_unit,*) ii,X1(i,1),X1(i,2),X1(i,3),V1(i,1),V1(i,2),
V1(i,3)
  enddo
  close(res_unit)
  open(unit=res_unit,file="stat.dat")
  read(res_unit,*) istart, nr,a
  read(res_unit,*) a,b,c
  read(res_unit,*)
  do i=1,Nbin
      read(res_unit,*) ii,gz1(i),sur1(i),pn1(i),pt1(i),a,b,c
  enddo
  close(res_unit)
!!$ Restarting the program from incomplete simuation data.
  END SUBROUTINE restart

  SUBROUTINE adjust_com
    real  :: com_z
    integer :: i,j
    com_z = 0
!!$  Adjusting com: center of mass in z direction
    do i = 1,Natm
       com_z = com_z + X(i,3)
    enddo
       com_z = 0.5*Lb(3) - com_z/Natm
       X(:,3) = X(:,3) + com_z
       do i = 1,Natm
          if (X(i,3).gt.Lb(3)) then
             X(i,3) = X(i,3) - Lb(3)
          else if (X(i,3).lt.0) then
             X(i,3) = X(i,3) + Lb(3)
          end if
       enddo
  END SUBROUTINE adjust_com

END PROGRAM run
!!!!
```

```
!! This is prms.f90
!! to define/read parameters
!!!!
MODULE prms

  IMPLICIT none

  integer              :: Natm ! number of atoms
  integer              :: Ntyp ! number of types of atoms
  integer,dimension(3) :: Nc   ! number of cells for PPPM
  integer,dimension(3) :: Ncc  ! number of cells for GAM_PPPM
  integer              :: Nt   ! number of timesteps
  real                 :: Ts   ! timestep
  real,dimension(3)    :: Lb   ! box size
  real,dimension(3)    :: H    ! cell space p3mlj (**computed**)
  real,dimension(3)    :: HH  ! cell space p3mgam (** computed**)
  real                 :: ranfac ! fractional change for
randomized pos
  real                 :: Qnh    ! Nose-Hoover thermostat mass
  real                 :: Ttar   ! target temperarture
  integer              :: Ts1    ! temperature control 1
  integer              :: Ts2    ! temperature control 2
  real                 :: tau1   ! Berendsen parameter 1
  real                 :: tau2   ! Berendsen parameter 2
  integer              :: Twal   ! regulate wall temperature ?
  real                 :: Ms     ! mass for feedback
temperature control
  integer              :: tmp_out ! temperature output interval
  integer              :: eng_out ! energy output interval
  integer              :: max_out ! Maxwellian output interval
  integer              :: rdf_out ! Radial dist. func. output
interval
  integer              :: btz_out ! Boltzamnn output interval
  integer              :: atm_out ! atom position output interval
  integer              :: ubr_out ! ubar output interval
  integer              :: res_out ! restart file output interval
  integer              :: mfx_out ! mean x force output
interval
  integer              :: Nmaxg   ! number of samples for
radial dist. func.
  integer              :: Nmaxw   ! number of samples in
```

```
1/2 Maxwellian
  real                 :: Vmax_peak ! peak Maxwellian velocity
(0 -- automatic)
  integer              :: Nbin   ! number of bins for density
and sigma calculation
  real                 :: ljcutoff ! LJ neighborlist cutoff how
many sigmax?
  real                 :: srcutoff ! SR neighborlist cutoff how
many sigmax?
  real                 :: listfac ! factor rl/rc for neighborlist
  integer              :: ntlist  ! how often to reconstruct the
neighborlist
  integer              :: Ntable  ! number of values in Fsr table
  integer              :: MaxNbr  ! Max number of neighbors
in list per atom
  integer              :: MaxNbrsr ! Max number of neighbors
in list per atom (PPPM Fsr)
  integer              :: I_whichic ! sets initial conditions (1
-- FCC solid channel )
  integer              :: I_time    ! sets which times agorithm (
1 -- Euler, 2 -- Verlet)
  real                 :: alpha     ! length parameter for PM
charge density distribution boundary
  real        :: sigmax  ! sigma for LJ
  integer, parameter :: prm_unit  =   5 !read unit for parameters
  integer, parameter :: out_unit  =   6 ! stdout
  integer, parameter :: atm_unit  =   7 ! atom locations
  integer, parameter :: max_unit  =   8 ! Maxwellian
  integer, parameter :: btz_unit  =   9 ! Boltzmann H
  integer, parameter :: tmp_unit  =  10 ! temperature
  integer, parameter :: stat_unit  =  11 ! statistics
  integer, parameter :: rdf_unit  =  12  ! radial dist function
  integer, parameter :: typ_unit  =  13  ! atom types
  integer, parameter :: ubr_unit  =  14  ! ubar
  integer, parameter :: res_unit  =  15  ! restart
  integer, parameter :: mfx_unit  =  16  ! mean x force on walls
  integer, parameter :: ran_unit  =  17  ! mean x force on walls
  real               :: Pi                         ! Pi (duh)
  real               :: srPi                       ! SQRT(Pi)
  real   , parameter :: kB     =   1.381E-23 ! k-Boltzmann(J/K)
  real   , parameter :: eps0   =   8.854187E-12 ! permittivity
```

```
constant (F/m = C^2/J m)
  real   , parameter :: ec     =   1.600219E-19 ! elementary
charge (C)

CONTAINS

  SUBROUTINE readprms ()

    real  :: alpha2

    Pi = 4.*ATAN(1.0)
    srPi = SQRT(Pi)
!!$ Reading input files

    read(prm_unit,*) Natm
    read(prm_unit,*) Ntyp

    read(prm_unit,*) Nc(1)
    read(prm_unit,*) Nc(2)
    read(prm_unit,*) Nc(3)
    read(prm_unit,*) Ncc(1)
    read(prm_unit,*) Ncc(2)
    read(prm_unit,*) Ncc(3)
    read(prm_unit,*) Nt
    read(prm_unit,*) Ts

    read(prm_unit,*) Lb(1)
    read(prm_unit,*) Lb(2)
    read(prm_unit,*) Lb(3)

    read(prm_unit,*) Ttar
    read(prm_unit,*) Ts1
    read(prm_unit,*) Ts2
    read(prm_unit,*) tau1
    read(prm_unit,*) tau2
    read(prm_unit,*) Twal
    read(prm_unit,*) Qnh
    read(prm_unit,*) pres
    read(prm_unit,*) Ms

    read(prm_unit,*) tmp_out
    read(prm_unit,*) eng_out
    read(prm_unit,*) max_out
    read(prm_unit,*) rdf_out
    read(prm_unit,*) btz_out
    read(prm_unit,*) atm_out
    read(prm_unit,*) ubr_out
    read(prm_unit,*) res_out
    read(prm_unit,*) mfx_out

    read(prm_unit,*) Nmaxg
    read(prm_unit,*) Nmaxw
    read(prm_unit,*) Vmax_peak
    read(prm_unit,*) Nbin
    read(prm_unit,*) ljcutoff
    read(prm_unit,*) srcutoff
    read(prm_unit,*) listfac
    read(prm_unit,*) ntlist
    read(prm_unit,*) Ntable
    read(prm_unit,*) MaxNbr
    read(prm_unit,*) MaxNbrsr
    read(prm_unit,*) alpha

    read(prm_unit,*) I_time
    read(prm_unit,*) I_whichic
    read(prm_unit,*) Alj1
    read(prm_unit,*) Clj1
    read(prm_unit,*) sigmax

    write(out_unit,*)
    write(out_unit,*) " ----- Physical paramters ----- "
    write(out_unit,*) "Natm    = ",Natm
    write(out_unit,*) "Ntyp    = ",Ntyp
    write(out_unit,*) "Nt      = ",Nt
    write(out_unit,*) "Ts      = ",Ts
    write(out_unit,*) "Lb(123) = ",Lb(1),Lb(2),Lb(3)
    write(out_unit,*) "Ttar    = ",Ttar
    write(out_unit,*) "Ts1     = ",Ts1
    write(out_unit,*) "Ts2     = ",Ts2
    write(out_unit,*) "tau1    = ",tau1
    write(out_unit,*) "tau2    = ",tau2
    write(out_unit,*) "Twal    = ",Twal
    write(out_unit,*) "Qnh     = ",Qnh
    write(out_unit,*) "pres    = ",pres
    write(out_unit,*) "Ms      = ",Ms
    write(out_unit,*) "Alj1      = ",Alj1
    write(out_unit,*) "Clj1      = ",Clj1
    write(out_unit,*) "sigmax     = ",sigmax

    write(out_unit,*)
    write(out_unit,*) " ------ Output paramters ----- "
    write(out_unit,*) "tmp_out   = ",tmp_out
    write(out_unit,*) "eng_out   = ",eng_out
    write(out_unit,*) "max_out   = ",max_out
    write(out_unit,*) "rdf_out   = ",rdf_out
    write(out_unit,*) "btz_out   = ",btz_out
    write(out_unit,*) "atm_out   = ",atm_out
    write(out_unit,*) "ubr_out   = ",ubr_out
    write(out_unit,*) "res_out   = ",res_out
    write(out_unit,*) "mfx_out   = ",mfx_out

    write(out_unit,*)
    write(out_unit,*) " ----- Maxwell paramters ----- "
    write(out_unit,*) "Nmaxg     = ",Nmaxg
    write(out_unit,*) "Nmaxw     = ",Nmaxw
    write(out_unit,*) "Vmax_peak = ",Vmax_peak
    write(out_unit,*)
    write(out_unit,*) " - Time Integration paramters - "
    write(out_unit,*) "I_time    = ",I_time
    write(out_unit,*) "I_whichic = ",I_whichic
    write(out_unit,*)
    write(out_unit,*) " ----- Ranging paramters ----- "
    write(out_unit,*) "ljcutoff  = ",ljcutoff
    write(out_unit,*) "srcutoff  = ",srcutoff
    write(out_unit,*) "listfac   = ",listfac
    write(out_unit,*) "Ntlist    = ",Ntlist
    write(out_unit,*) "Ntable    = ",Ntable
    write(out_unit,*) "MaxNbr    = ",MaxNbr
    write(out_unit,*) "MaxNbrsr  = ",MaxNbrsr
    write(out_unit,*) "Nc        = ",Nc(:)
    write(out_unit,*) "alpha     = ",alpha
```

```
!!$ Compute cell spaceing
    H = Lb/REAL(Nc)
    HH = Lb/REAL(Ncc)
    write(out_unit,*)
    write(out_unit,*) "H        = ",H(1),H(2),H(3)
    write(out_unit,*) "HH       = ",HH(1),HH(2),HH(3)
    write(out_unit,*) "alpha    = ",alpha

  END SUBROUTINE readprms

END MODULE prms
!!!!
!! This is data.f90
!! to define/intialize the systems
!!!!
MODULE data

  USE prms


  IMPLICIT none

  real, allocatable, dimension(:,:)   :: X   ! atomic positions
  real, allocatable, dimension(:,:)   :: V   ! atomic velocities
  CONTAINS
  SUBROUTINE initdata
!!$ Intializing the atomic positions and velocities
    real :: ran,v2,sumv2,fs !tmp variables
    integer     :: i1,j1,k1,i,j,nx,ny,nz,nxny
    real        :: min_d
    real :: seed,uni
    real,dimension(3) :: xr(3),sumV(3)
      allocate (X(Natm,3),V(Natm,3))
      ran = uni(100)
      min_d = 1.3
      sumV(:) = 0
      nx = int(Lb(1)/min_d)
      ny = int(Lb(2)/min_d)
      nxny = nx*ny
      nz = int((Natm-1)/nxny)+1
      do i = 1,Natm
```

```
         k1 = (i - 1)/nxny
         j1 = (i - nxny*k1 - 1)/nx
         i1 = i - nxny*k1 - nx*j1
         X(i,1) = (i1 - 1)*min_d
         X(i,2) = j1*min_d
         X(i,3) = .5*Lb(3) - (nz/2)*min_d + k1*min_d
         V(i,1) = uni(0) - 0.5
         V(i,2) = uni(0) - 0.5
         V(i,3) = uni(0) - 0.5
         sumV(:) = sumV(:) + V(i,:)
      enddo
    sumv2 = 0
    do i = 1,Natm
       V(i,:) = V(i,:) - sumV(:)/Natm
       v2 = V(i,1)**2 + V(i,2)**2 + V(i,3)**2
       sumv2 = sumv2 + v2
    enddo
!!$ Rescaling the velocities to the desired temperature
    fs = sqrt(3*Ttar*Natm/sumv2)
    do i = 1,Natm
       V(i,:) = V(i,:) * fs
!      write(*,*) "X & V(",i,")",X(i,:),V(i,:)
    enddo
    END SUBROUTINE initdata

END MODULE data
!!!!
!! This is nlistmod.f90
!! to find head-of-chain and atom's cell position
!!!!
MODULE nlistmod

  USE prms
  USE data

  IMPLICIT none

CONTAINS

  SUBROUTINE atomcell(X,Icell,hspace,Nmax)
    real,dimension(Natm,3)       :: X      !particle positions
    integer,dimension(Natm,3)    :: Icell  !atom cell list
    real,dimension(3)            :: hspace !mesh cell spacing for
distribution
    integer,dimension(3)         :: Nmax     ! max cell
    integer                      :: l
    integer i

    do l = 1,3
       Icell(:,l) = MAX(MIN(INT(X(:,l)/hspace(l)) + 1,Nmax(l)),1)
    end do
!!$ Detecting the cell number of the atoms
  END SUBROUTINE atomcell

  SUBROUTINE chainlist(LL,HOC,N,X)
!!$ Creating chainlist for short range calculation
    integer,dimension(Natm) :: LL    ! next in list pointer
    integer,dimension(3)    :: N     ! either Ncsr or Nclj
    integer,dimension(0:N(1)+1,0:N(2)+1,0:N(3)+1) :: HOC
! Head-of-Chain pointer with perioidic continuation
    real,dimension(Natm,3)  :: X     ! particle positions
    integer,dimension(Natm,3):: Icell ! atom cell list
    integer                 :: i
    real,dimension(3)       :: HMf   ! cell spacing

    HMf = Lb/REAL(N)
    call atomcell(X,Icell,HMf,N)

    HOC = 0
    do i = 1,Natm
       LL(i) = HOC(Icell(i,1),Icell(i,2),Icell(i,3))
       HOC(Icell(i,1),Icell(i,2),Icell(i,3)) = i
    end do

    HOC(0,:,:)        = HOC(N(1),:,:)
    HOC(N(1)+1,:,:)   = HOC(1,:,:)
    HOC(:,0,:)        = HOC(:,N(2),:)
    HOC(:,N(2)+1,:)   = HOC(:,1,:)
    HOC(:,:,0)        = HOC(:,:,N(3))
    HOC(:,:,N(3)+1)   = HOC(:,:,1)

  END SUBROUTINE chainlist
END MODULE nlistmod
!!!!
!! This is p3mlj.f90
!! to calculate the force using P3M method
!!!!
MODULE p3mlj

  USE prms
  USE nlistmod
  USE data

!!$ This is p3m code for force evaluation

  IMPLICIT none

  integer(8)   :: plan_r2c        ! plan for forward transforms
  integer(8)   :: plan_c2r        ! plan for inverse transforms

! NOTE THESE SHOULD HAVE THE SIZE OF A GCC POINTER.
!8 ON ALPHA, 4 ON INTEL

  real, allocatable, dimension(:,:,:)  :: Ghat
! influence function
  real, allocatable, dimension(:)      :: kx,ky,kz
! all wavenumbers
  real, allocatable, dimension(:)      :: Fsrtable
! lookuptable for Fsr
  TYPE :: listelm
     integer(1), dimension(3) :: offset
! integer periodic offset
     integer                  :: nbr              ! the neighbor
  END TYPE listelm

  TYPE(listelm), allocatable, dimension(:) :: list
  integer, allocatable,dimension(:,:)      :: nlist

  integer,dimension(3) :: Ncsr  ! number of cells for SR
neighborlist (**computed**)
  real                :: rc_sr ! cuttoff for Fsr (**computed**)
```

```
CONTAINS


  SUBROUTINE LJ_forcePPPM(X,q,F)
    real,dimension(Natm,3)    :: X       ! particle positions
    real,dimension(Natm)       :: q       ! atomic charges
    real,dimension(Natm,3)    :: F,F1,F2       ! force (xyz)
    real,dimension(Natm,3)    :: R       ! PM force
!!$ Calculating total force on atom,R: long range,F: short range
    call LJ_computeR(X,q,R)
    call LJ_computeFsr(X,q,F)
    F1 = -F
    F2 = -R
    F = 4.0*(-  F -  R)

  END SUBROUTINE LJ_forcePPPM


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Interpolation/distribution of P to M and M to P

  SUBROUTINE LJ_weights(X,Icell,Wgts)
    real,dimension(Natm,3)       :: X       ! particle positions
    integer,dimension(Natm,3)    :: Icell   ! atom cell list
    real,dimension(-1:1,-1:1,-1:1,Natm)   :: Wgts
! interpolation/distribution weights
    real,dimension(3)                     ::  Xp
! distance of particle from cell center
    real,dimension(3,-1:1)                ::  w
! weights for distributing charge

    real                                  ::  T1,T2
    integer                               ::  it1,it2,it3
    integer                               ::  i


    do i = 1,Natm
       Xp = (X(i,:) - (H(:)*REAL(Icell(i,:)-1) + 0.5*H(:)))/H(:)
       w(:, 1) = 0.5*(0.5+Xp(:))*(0.5+Xp(:))
       w(:, 0) = 0.75 - Xp(:)*Xp(:)
       w(:,-1) = 0.5*(0.5-Xp(:))*(0.5-Xp(:))
```

```
      do it1 = -1,1
         T1 = w(1,it1)
         do it2 = -1,1
            T2 = T1*w(2,it2)
            do it3 = -1,1
               Wgts(it3,it2,it1,i) = T2*w(3,it3)
            end do
         end do
      end do
   end do


  END SUBROUTINE LJ_weights

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Interpolation/distribution of P to M and M to P

  SUBROUTINE LJ_distribute(Icell,q,Wgts,rho)
    integer,dimension(Natm,3)       :: Icell    ! atom cell list
    real,dimension(Natm)            :: q        ! atomic charges
    real,dimension(Nc(1),Nc(2),Nc(3))   :: rho
! mesh charge density
    real,dimension(-1:1,-1:1,-1:1,Natm) :: Wgts
! interpolation/distribution weights
    real,dimension(0:Nc(1)+1,0:Nc(2)+1,0:Nc(3)+1)  ::  rhow
! accumulating charge with xtra rows

    integer                                :: it1,it2,it3
    integer                                :: i

    rhow = 0.

    do i = 1,Natm
       do it1 = -1,1
          do it2 = -1,1
             do it3 = -1,1
     rhow(Icell(i,1)+it1, Icell(i,2)+it2, Icell(i,3)+it3) =   &
        + rhow(Icell(i,1)+it1, Icell(i,2)+it2, Icell(i,3)+it3) &
                  +  q(i)*Wgts(it3,it2,it1,i)
             end do
          end do
       end do
    end do

    rhow(1,:,:)     = rhow(1,:,:)     + rhow(Nc(1)+1,:,:)
    rhow(Nc(1),:,:) = rhow(Nc(1),:,:) + rhow(0,:,:)
    rhow(:,1,:)     = rhow(:,1,:)     + rhow(:,Nc(2)+1,:)
    rhow(:,Nc(2),:) = rhow(:,Nc(2),:) + rhow(:,0,:)
    rhow(:,:,1)     = rhow(:,:,1)     + rhow(:,:,Nc(3)+1)
    rhow(:,:,Nc(3)) = rhow(:,:,Nc(3)) + rhow(:,:,0)

    rho =  rhow(1:Nc(1),1:Nc(2),1:Nc(3))

  END SUBROUTINE LJ_distribute

  SUBROUTINE LJ_interpolate(Icell,F,Wgts,R,N)
!!$ Interpolate the field from M to P position
    integer                     :: N     ! usually 3 for vector
    integer,dimension(Natm,3)   :: Icell ! atom cell list
    real,dimension(Natm,N)      :: R     ! Force (xyz) on atom
    real,dimension(Nc(1),Nc(2),Nc(3),N) :: F   ! PM force (xyz)
    real,dimension(-1:1,-1:1,-1:1,Natm) :: Wgts
! interpolation/distribution weights
    real,dimension(0:Nc(1)+1,0:Nc(2)+1,0:Nc(3)+1,N)  ::  Fw
! PM force (store xtra rows)

    integer                               ::  it1,it2,it3
    integer                               ::  i,l

    Fw(1:Nc(1),1:Nc(2),1:Nc(3),:) = F
    Fw(0,:,:,:)       = Fw(Nc(1),:,:,:)
    Fw(Nc(1)+1,:,:,:) = Fw(1,:,:,:)
    Fw(:,0,:,:)       = Fw(:,Nc(2),:,:)
    Fw(:,Nc(2)+1,:,:) = Fw(:,1,:,:)
    Fw(:,:,0,:)       = Fw(:,:,Nc(3),:)
    Fw(:,:,Nc(3)+1,:) = Fw(:,:,1,:)
    R = 0.

    do l = 1,3
       do i = 1,Natm
          do it1 = -1,1
```

```
             do it2 = -1,1
                do it3 = -1,1
                   R(i,l) = R(i,l) &
           + Fw(Icell(i,1)+it1,Icell(i,2)+it2,Icell(i,3)+it3,l)
*Wgts(it3,it2,it1,i)
                end do
             end do
          end do
       end do
    end do

  END SUBROUTINE LJ_interpolate

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  SHORT RANGE FORCES
  SUBROUTINE LJ_computeFsr(X,q,Fsr)
    real, dimension(Natm,3) :: X
    real, dimension(Natm,3) :: Fsr
    real, dimension(Natm)   :: q

    real    :: rij1,rij2,fc
    real    :: fac1,fac2,r2,src2
    real, dimension(3)    :: xx,ff
    integer :: i,j,k,it

    integer l

    Fsr = 0.
    src2 = srcutoff*srcutoff
    fac1 = REAL(Ntable)/(rc_sr**2*listfac**2)
    fac2 = 1./fac1

    do i = 1,Natm
        do k = nlist(i,1),nlist(i,2)
         j = list(k)%nbr
             xx(:) = X(i,:) - X(j,:)
             xx(:) = xx(:) - Lb(:)*nint(xx(:)/Lb(:))
          rij2 = (xx(1)*xx(1) + xx(2)*xx(2) + xx(3)*xx(3))
          if (rij2 .lt. src2) then
           it = INT(rij2*fac1)
           r2 = REAL(it)*fac2
        fc =  q(i)*q(j)* ((Fsrtable(it+1)-Fsrtable(it))*fac1*
(rij2-r2) + Fsrtable(it))
           ff  = fc*xx
           Fsr(i,:) = Fsr(i,:) + ff
           Fsr(j,:) = Fsr(j,:) - ff
          end if
       end do
    end do

  END SUBROUTINE LJ_computeFsr

  SUBROUTINE LJ_makeFsrtable
!!$ For faster evaluation, short range table is used as r^2
    integer    :: i
    real    :: r2,r,a2,r7,gprime,gp,ar2

    a2 = alpha*alpha
    do i = 1,Ntable
       r2 = REAL(i)/REAL(Ntable)*rc_sr*rc_sr*listfac*listfac
       r = SQRT(r2)
       ar2 = a2*r2
       gp = EXP(-ar2)*(1. + ar2 + ar2*ar2/2.)
       r7 = r2*r2*r2*r

       gprime = 2.*alpha*r*((1 + ar2)*EXP(-ar2) - gp)

       Fsrtable(i) = 1./r7*( 6.*gp/r - alpha*gprime)-12/(r7*r7)

    end do

  END SUBROUTINE LJ_makeFsrtable

  SUBROUTINE LJ_nlistFsr(X)
!!$ Making a neighbor list for Fsr calculation

    real   ,dimension(Natm,3)     :: X    ! particle positions
    integer,dimension(Natm)       :: LL   ! next in list pointer
    integer,dimension(0:Ncsr(1)+1,0:Ncsr(2)+1,0:Ncsr(3)+1) :: HOC
! H-of-C pointer with perioidic continuation
    real,dimension(3)     :: xx
    integer               :: ic1,ic2,ic3
    real                 :: rij2,fac2
    integer              :: i,j,n,l
    integer              :: nbr_cnt
    integer, dimension(3) :: ndx
    integer,dimension(3*13), parameter :: poff = (/ 1 , 0 , 0 , &
! offsets for next cells
                                                  1 , 0 , 1 , &
                                                  1 , 0 ,-1 , &
                                                  1 , 1 , 0 , &
                                                  1 , 1 , 1 , &
                                                  1 , 1 ,-1 , &
                                                  1 ,-1 , 0 , &
                                                  1 ,-1 , 1 , &
                                                  1 ,-1 ,-1 , &
                                                  0 , 1 , 1 , &
                                                  0 , 1 , 0 , &
                                                  0 , 1 ,-1 , &
                                                  0 , 0 , 1  /)

    call chainlist(LL,HOC,Ncsr,X)
10  continue

    nbr_cnt = 1
    fac2 = rc_sr*rc_sr*listfac*listfac
    do ic3 = 1,Ncsr(3)
       do ic2 = 1,Ncsr(2)
          do ic1 = 1,Ncsr(1)
             i = HOC(ic1,ic2,ic3)  ! start with atom at the head
of the linked list
             do while(i.ne.0)      ! loop over all atoms in the
linked list if any
                nlist(i,1) = nbr_cnt

                ! IN CELL
                j = LL(i)      ! start with the next atom in the
linked list if any

                do while(j.gt.0)
                   xx = X(i,:) - X(j,:)
                   rij2 = xx(1)*xx(1)+xx(2)*xx(2)+xx(3)*xx(3)
! rij2 must be less than the cutoff
```

```
                   if (rij2.lt.fac2) then
                      list(nbr_cnt)%offset(:) = 0
! periodic offset factor
                      list(nbr_cnt)%nbr = j
! neighbor index
                      nbr_cnt = nbr_cnt + 1
                      if (nbr_cnt.gt.MaxNbrsr) then
                         goto 20
                      endif

                   end if

                   j = LL(j)

                end do

              do n = 1,39,3  ! loop over 13 other cells
                 j = HOC(ic1+poff(n),ic2+poff(n+1),ic3+poff(n+2))
                 do while(j.ne.0)

                      xx = X(i,:) - X(j,:)
                      ndx = 0
                      do l = 1,3
                         if (xx(l).gt.Lb(l)/2.) then
                            xx(l) = xx(l) - Lb(l)
                            ndx(l) = -1
                         else if (xx(l).lt.-Lb(l)/2.) then
                            xx(l) = xx(l) + Lb(l)
                            ndx(l) = +1
                         end if
                      end do

                   rij2 = xx(1)*xx(1)+xx(2)*xx(2)+xx(3)*xx(3)

             ! rij2 must be less than the cutoff
                   if (rij2.lt.fac2) then

                     list(nbr_cnt)%offset(:) = ndx    ! periodic
offset factor
                     list(nbr_cnt)%nbr = j            ! neighbor
index
```

```
                         nbr_cnt = nbr_cnt + 1
                      if (nbr_cnt.gt.MaxNbrsr) then
                         goto 20
                      endif

                      end if

                      j = LL(j)

                   end do

                end do

                nlist(i,2) = nbr_cnt - 1
                i = LL(i)
             end do

          end do
       end do
    end do
20  continue
!!$ For right size neighbor list
    if (nbr_cnt.gt.MaxNbrsr) then
       write(*,*) "nbr_cnt,MaxNbrsr",nbr_cnt,MaxNbrsr
       deallocate(list)
       MaxNbrsr = INT(1.5*nbr_cnt) + 1
       allocate(list(MaxNbrsr))
       goto 10
    else if (MaxNbrsr.gt.1.35*nbr_cnt.and.MaxNbrsr.gt.100) then
       deallocate(list)
       MaxNbrsr = INT(1.25*nbr_cnt)+1
       allocate(list(MaxNbrsr))
       write(out_unit,*)"Lowering MaxNbrsr to ",MaxNbrsr
       goto 10
    else if (MaxNbrsr.lt.1.05*nbr_cnt.and.MaxNbrsr.gt.100) then
       deallocate(list)
       MaxNbrsr = INT(1.25*nbr_cnt)+1
       allocate(list(MaxNbrsr))
       write(out_unit,*)"Raising MaxNbrsr to ",MaxNbrsr
       goto 10
    end if
```

```
!!$    print *,"Fsr nbr_cnt",nbr_cnt

  END SUBROUTINE LJ_nlistFsr

  SUBROUTINE LJ_computeR(X,q,R)
    real,dimension(Natm,3)    :: X        ! particle positions
    real,dimension(Natm)      :: q        ! atomic charges
    real,dimension(Natm,3)    :: R        ! PM force (xyz)

    integer,dimension(Natm,3)         :: Icell  ! atom cell list
    real,dimension(Nc(1),Nc(2),Nc(3)) :: rho    ! mesh charge
density
    real,dimension(Nc(1),Nc(2),Nc(3),3) :: F      ! mesh force
(xyz)
    real,dimension(-1:1,-1:1,-1:1,Natm) :: Wgts   ! interpolation
/distribution weights

    integer                   :: i

    call atomcell(X,Icell,H,Nc)
    call LJ_weights(X,Icell,Wgts)
    call LJ_distribute(Icell,q,Wgts,rho)

    call LJ_poisson(F,rho)

    call LJ_interpolate(Icell,F,Wgts,R,3)

! multiply by charge to compute force at each atom
    do i = 1,Natm
       R(i,:) = R(i,:)*q(i)
    end do

  END SUBROUTINE LJ_computeR

  SUBROUTINE LJ_poisson(F,rho)
    real,dimension(Nc(1),Nc(2),Nc(3),3) :: F      ! force (xyz)
    real,dimension(Nc(1),Nc(2),Nc(3))   :: rho
! mesh charge density
    complex,dimension(Nc(1)/2+1,Nc(2),Nc(3))    :: rhohat
! transformed source
```

```
    complex,dimension(Nc(1)/2+1,Nc(2),Nc(3),3)  :: fhat
! transformed force

    integer          ::  k1,k2,k3

! Solve "Poisson" -- multiply by optimal influence function
    call rfftwnd_f77_one_real_to_complex(plan_r2c,rho,rhohat)
    rhohat = -rhohat*Ghat

! Compute force (multiply by wave number i kx,i ky,i kz)
    do k3 = 1,Nc(3)
       do k2 = 1,Nc(2)
          do k1 = 1,Nc(1)/2+1
             fhat(k1,k2,k3,1) = rhohat(k1,k2,k3)*(0.,1.)*kx(k1)
             fhat(k1,k2,k3,2) = rhohat(k1,k2,k3)*(0.,1.)*ky(k2)
             fhat(k1,k2,k3,3) = rhohat(k1,k2,k3)*(0.,1.)*kz(k3)
          end do
       end do
    end do

! Inverse transform potential and forces
    call rfftwnd_f77_one_complex_to_real(plan_c2r,fhat(1,1,1,1)
,F(1,1,1,1))
    call rfftwnd_f77_one_complex_to_real(plan_c2r,fhat(1,1,1,2)
,F(1,1,1,2))
    call rfftwnd_f77_one_complex_to_real(plan_c2r,fhat(1,1,1,3)
,F(1,1,1,3))

  END SUBROUTINE LJ_poisson

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FFT stuff
  SUBROUTINE LJ_initPPPM

    allocate (Ghat(Nc(1)/2+1,Nc(2),Nc(3)))
    allocate (list(MaxNbrsr),nlist(Natm,2))
    allocate (Fsrtable(Ntable))

    rc_sr = srcutoff*sigmax
    Ncsr = MAX(1,INT(Lb/rc_sr))
    print *,"Ncsr = ",Ncsr
  END SUBROUTINE LJ_initPPPM

  SUBROUTINE LJ_newalpha
!!$ Only once, create Ghat and Fsr table

    call LJ_computeGhat
    call LJ_makeFsrtable

  END SUBROUTINE LJ_newalpha

  SUBROUTINE LJ_computeGhat

    integer    :: k1,k2,k3,kk2,kk3

    do k1 = 0,Nc(1)/2
       do k2 = 0,Nc(2)/2
          do k3 = 0,Nc(3)/2
      if (k1.ne.0 .or. k2.ne.0 .or. k3.ne.0) call LJ_GetGhat(k1,
k2,k3,Ghat(k1+1,k2+1,k3+1))
          end do
          do k3 = Nc(3)/2+1,Nc(3)-1
             kk3 = Nc(3)-k3
      if (k1.ne.0 .or. k2.ne.0 .or. kk3.ne.0) call LJ_GetGhat(
k1,k2,kk3,Ghat(k1+1,k2+1,k3+1))
          end do
       end do
       do k2 = Nc(2)/2+1,Nc(2)-1
          kk2 = Nc(2)-k2
          do k3 = 0,Nc(3)/2
      if (k1.ne.0 .or. kk2.ne.0 .or. k3.ne.0) call LJ_GetGhat(
k1,kk2,k3,Ghat(k1+1,k2+1,k3+1))
          end do
          do k3 = Nc(3)/2+1,Nc(3)-1
             kk3 = Nc(3)-k3
      if (k1.ne.0 .or. kk2.ne.0 .or. kk3.ne.0) call LJ_GetGhat(
k1,kk2,kk3,Ghat(k1+1,k2+1,k3+1))
          end do
       end do
    end do
    Ghat(1,1,1) = 0.
```

```
!    Ghat = -Ghat/(4.*Pi)/eps0/(Lb(1)*Lb(2)*Lb(3))
    Ghat = -Ghat/(Lb(1)*Lb(2)*Lb(3))

  END SUBROUTINE LJ_computeGhat

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FFT stuff
  SUBROUTINE LJ_initfft
!!$ Initiating the FFT plan using FFTW subroutine

    integer    :: k1,k2,k3

    allocate (kx(Nc(1)/2+1),ky(Nc(2)),kz(Nc(3)))

    call rfftw3d_f77_create_plan(plan_c2r,Nc(1),Nc(2),Nc(3),+1,
0) !1)  ! comp_to_real
    call rfftw3d_f77_create_plan(plan_r2c,Nc(1),Nc(2),Nc(3),-1,
0) !1)  ! real_to_comp

!  Compute wave numbers for xyz derivatives
    do k1 = 0,Nc(1)/2
       kx(k1+1) = REAL(k1)*2.*Pi/Lb(1)
    end do
    do k2 = 0,Nc(2)/2
       ky(k2+1) = REAL(k2)*2.*Pi/Lb(2)
    end do
    do k2 = Nc(2)/2+1,Nc(2)-1
       ky(k2+1) = -REAL(Nc(2)-k2)*2.*Pi/Lb(2)
    end do
    do k3 = 0,Nc(3)/2
       kz(k3+1) = REAL(k3)*2.*Pi/Lb(3)
    end do
    do k3 = Nc(3)/2+1,Nc(3)-1
       kz(k3+1) = -REAL(Nc(3)-k3)*2.*Pi/Lb(3)
    end do

  END SUBROUTINE LJ_initfft

  SUBROUTINE LJ_getU2sum(k1,k2,k3,U2)
!!$ Get sum(U^2)
    integer  :: k1,k2,k3
    real  :: rk1,rk2,rk3,U2

    rk1 = REAL(k1)*2.*Pi/Lb(1)
    rk2 = REAL(k2)*2.*Pi/Lb(2)
    rk3 = REAL(k3)*2.*Pi/Lb(3)

    U2 =   (1. - SIN(rk1*H(1)/2.)**2 + 2./15.*SIN(rk1*H(1)/2.)
**4) &
         * (1. - SIN(rk2*H(2)/2.)**2 + 2./15.*SIN(rk2*H(2)/2.)*
*4) &
         * (1. - SIN(rk3*H(3)/2.)**2 + 2./15.*SIN(rk3*H(3)/2.)*
*4)

  END SUBROUTINE LJ_getU2sum

  SUBROUTINE LJ_getRU2sum(k1,k2,k3,RU2)
!!$ Get sum(R*U^2)
    integer  :: k1,k2,k3
    integer  :: m1,m2,m3
    real  :: rk1,rk2,rk3
    real  :: rm1,rm2,rm3
    real, dimension(3)  :: R,RU2
    real                :: a,b,c,f1,f2,f3
    integer, parameter :: Mmax=4

    rk1 = REAL(k1)*2.*Pi/Lb(1)
    rk2 = REAL(k2)*2.*Pi/Lb(2)
    rk3 = REAL(k3)*2.*Pi/Lb(3)

    RU2 =0.
    do m1 = -Mmax,Mmax

       rm1 = REAL(m1)
       a = rk1+2.*Pi*rm1/H(1)
       if (a.eq.0) then
          f1 = 1.
       else
          f1 = SIN(a*H(1)/2.)/(a*H(1)/2.)
       end if

       do m2 = -Mmax,Mmax
```

```
          rm2 = REAL(m2)
          b = rk2+2.*Pi*rm2/H(2)
          if (b.eq.0) then
             f2 = 1.
          else
             f2 = SIN(b*H(2)/2.)/(b*H(2)/2.)
          end if

          do m3 = -Mmax,Mmax

             rm3 = REAL(m3)
             c = rk3+2.*Pi*rm3/H(3)
             if (c.eq.0) then
                f3 = 1.
             else
                f3 = SIN(c*H(3)/2.)/(c*H(3)/2.)
             end if

             call LJ_getR(a,b,c,R)
             RU2 = RU2  + R*(f1*f2*f3)**6

          end do
       end do
    end do

  END SUBROUTINE LJ_getRU2sum

  SUBROUTINE LJ_getGhat(k1,k2,k3,Ghat)
!!$ Get Ghat_optimized
    integer  :: k1,k2,k3
    real  :: Ghat,U2
    real, dimension(3)  :: RU2,D

    call LJ_getRU2sum(k1,k2,k3,RU2)
    call LJ_getU2sum(k1,k2,k3,U2)
    call LJ_getD(k1,k2,k3,D)

    Ghat = (D(1)*RU2(1) + D(2)*RU2(2) + D(3)*RU2(3) )/ &
         ( (D(1)*D(1) + D(2)*D(2) + D(3)*D(3))*U2*U2 )
  END SUBROUTINE LJ_getGhat

  SUBROUTINE LJ_getD(k1,k2,k3,D)
!!$ Diffrential operator
    integer  :: k1,k2,k3
    real  :: rk1,rk2,rk3
    real,dimension(3) :: D

    rk1 = REAL(k1)*2.*Pi/Lb(1)
    rk2 = REAL(k2)*2.*Pi/Lb(2)
    rk3 = REAL(k3)*2.*Pi/Lb(3)

    D(1) = rk1 ; D(2) = rk2 ; D(3) = rk3

  END SUBROUTINE LJ_getD

  SUBROUTINE LJ_getR(rk1,rk2,rk3,R)
!!$ Get R(k) in Fourier space
    real  :: rk1,rk2,rk3
    real,dimension(3) :: R

    real  :: ksq,f,kp,Yc,f1,f2,a2,k1

    real  :: DERF

    ksq = (rk1*rk1 + rk2*rk2 + rk3*rk3)
    k1 = SQRT(ksq)
    kp = SQRT(rk1*rk1+rk3*rk3)
    Yc = Lb(2)*.5
    a2 = alpha*alpha

    if (ksq.gt.0) then

       f1 = Pi*srPi*alpha*a2/3.*((1.-ksq/2./a2)*EXP(-ksq/4./a2)
+ksq*k1/4./a2/alpha*srPi*(1.-ERF(k1/2./alpha)))
       f2 = 1.
!!$ f1 is same as f_p(k/2beta)
!!$ Multiply by -i k_x for refernce force
       R(1) = -rk1*f1*f2  ! leaving out i
       R(2) = -rk2*f1*f2  ! leaving out i
       R(3) = -rk3*f1*f2  ! leaving out i
    else

       R = 1.

    end if

  END SUBROUTINE LJ_getR

  SUBROUTINE LJ_endfft

     call rfftwnd_f77_destroy_plan(plan_r2c)
     call rfftwnd_f77_destroy_plan(plan_c2r)

   END SUBROUTINE LJ_endfft

END MODULE p3mlj

!!!!
!! This is p3mgam.f90
!! to calculate the surface tension using P3M method
!!!!
MODULE p3mgam

  USE prms
  USE nlistmod
  USE data

!!$ Coments are same as p3mlj module, follow that for more
explation
!!$ P3M code for surface tension evaluation
  IMPLICIT none

  integer(8)   :: plan_r2c   ! plan for forward transforms
  integer(8)   :: plan_c2r   ! plan for inverse transforms
! NOTE THESE SHOULD HAVE THE SIZE OF
! A GCC POINTER. 8 ON ALPHA, 4 ON INTEL
  real, allocatable, dimension(:,:,:)  :: Ghat
! influence function
  real, allocatable, dimension(:)      :: kx,ky,kz
! all wavenumbers
```

```
  real, allocatable, dimension(:)      :: Fsrtable
! lookuptable for Fsr

!!$$ The following variables are defined in the nlistmod module.
removing from the p3mlj also.

  TYPE :: listelm
     integer(1), dimension(3) :: offset
! integer periodic offset
     integer                  :: nbr           ! the neighbor
  END TYPE listelm
  TYPE(listelm), allocatable, dimension(:) :: list
  integer, allocatable,dimension(:,:)      :: nlist
  integer,dimension(3) :: Ncsr
! number of cells for SR neighborlist (**computed**)
  real                 :: rc_sr
! cuttoff for Fsr (**computed**)

CONTAINS


  SUBROUTINE GAM_forcePPPM(X,q,F,pn,pt,gamF,gamR)
    real,dimension(Natm,3)    :: X      ! particle positions
    real,dimension(Natm)      :: q      ! atomic charges
    real,dimension(Natm)  :: gamR,gamF,F,pn,pt,pnF,ptF,pnR,ptR
! PM force
    real :: area

    call GAM_computeR(X,q,gamR,pnR,ptR)
    call GAM_computeFsr(X,q,gamF,pnF,ptF)
    area = Lb(1)*Lb(2)
    gamF = gamF/area
    gamR = gamR/area
    F = gamF + gamR
    pn = pnF + pnR
    pt = ptF + ptR
    pn = pn/area
    pt = pt/area

  END SUBROUTINE GAM_forcePPPM
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Interpolation/distribution of P to M and M to P

  SUBROUTINE GAM_weights(X,Icell,Wgts)
    real,dimension(Natm,3)      :: X       ! particle positions
    integer,dimension(Natm,3)   :: Icell   ! atom cell list
    real,dimension(-1:1,-1:1,-1:1,Natm)   :: Wgts
! interpolation/distribution weights
    real,dimension(3)                     ::  Xp
! distance of particle from cell center
    real,dimension(3,-1:1)                ::  w
! weights for distributing charge

    real                                  ::  T1,T2
    integer                               ::  it1,it2,it3
    integer                               ::  i


    do i = 1,Natm
       Xp = (X(i,:)-(HH(:)*REAL(Icell(i,:)-1)+0.5*HH(:)))/HH(:)
       w(:, 1) = 0.5*(0.5+Xp(:))*(0.5+Xp(:))
       w(:, 0) = 0.75 - Xp(:)*Xp(:)
       w(:,-1) = 0.5*(0.5-Xp(:))*(0.5-Xp(:))

       do it1 = -1,1
          T1 = w(1,it1)
          do it2 = -1,1
             T2 = T1*w(2,it2)
             do it3 = -1,1
                Wgts(it3,it2,it1,i) = T2*w(3,it3)
             end do
          end do
       end do
    end do


  END SUBROUTINE GAM_weights

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Interpolation/distribution of P to M and M to P
```

```
  SUBROUTINE GAM_distribute(Icell,q,Wgts,rho)
    integer,dimension(Natm,3)     :: Icell    ! atom cell list
    real,dimension(Natm)          :: q        ! atomic charges
    real,dimension(Ncc(1),Ncc(2),Ncc(3))   :: rho
! mesh charge density
    real,dimension(-1:1,-1:1,-1:1,Natm) :: Wgts
! interpolation/distribution weights
    real,dimension(0:Ncc(1)+1,0:Ncc(2)+1,0:Ncc(3)+1)  ::  rhow
! accumulating charge with xtra rows
    integer                               ::  it1,it2,it3
    integer                               ::  i

    rhow = 0.

    do i = 1,Natm
       do it1 = -1,1
          do it2 = -1,1
             do it3 = -1,1
      rhow(Icell(i,1)+it1, Icell(i,2)+it2, Icell(i,3)+it3) =   &
       +  rhow(Icell(i,1)+it1, Icell(i,2)+it2, Icell(i,3)+it3) &
       +  q(i)*Wgts(it3,it2,it1,i)
             end do
          end do
       end do
    end do

    rhow(1,:,:)     = rhow(1,:,:)     + rhow(Ncc(1)+1,:,:)
    rhow(Ncc(1),:,:) = rhow(Ncc(1),:,:) + rhow(0,:,:)
    rhow(:,1,:)     = rhow(:,1,:)     + rhow(:,Ncc(2)+1,:)
    rhow(:,Ncc(2),:) = rhow(:,Ncc(2),:) + rhow(:,0,:)
    rhow(:,:,1)     = rhow(:,:,1)     + rhow(:,:,Ncc(3)+1)
    rhow(:,:,Ncc(3)) = rhow(:,:,Ncc(3)) + rhow(:,:,0)

    rho =  rhow(1:Ncc(1),1:Ncc(2),1:Ncc(3))

  END SUBROUTINE GAM_distribute

  SUBROUTINE GAM_interpolate(Icell,F,Wgts,R,N)
    integer                   :: N     ! usually 3 for vector
    integer,dimension(Natm,3) :: Icell ! atom cell list
    real,dimension(Natm,N)    :: R     ! Force (xyz) on atom
    real,dimension(Ncc(1),Ncc(2),Ncc(3),N) :: F ! PM force (xyz)
    real,dimension(-1:1,-1:1,-1:1,Natm) :: Wgts ! interpolation
/distribution weights
    real,dimension(0:Ncc(1)+1,0:Ncc(2)+1,0:Ncc(3)+1,N)  ::
Fw   ! PM force (store xtra rows)
    integer                               ::  it1,it2,it3
    integer                               ::  i,l

    Fw(1:Ncc(1),1:Ncc(2),1:Ncc(3),:) = F
    Fw(0,:,:,:)       = Fw(Ncc(1),:,:,:)
    Fw(Ncc(1)+1,:,:,:) = Fw(1,:,:,:)
    Fw(:,0,:,:)       = Fw(:,Ncc(2),:,:)
    Fw(:,Ncc(2)+1,:,:) = Fw(:,1,:,:)
    Fw(:,:,0,:)       = Fw(:,:,Ncc(3),:)
    Fw(:,:,Ncc(3)+1,:) = Fw(:,:,1,:)
    R = 0.

    do l = 1,3
       do i = 1,Natm
          do it1 = -1,1
             do it2 = -1,1
                do it3 = -1,1
                   R(i,l) = R(i,l) &
           + Fw(Icell(i,1)+it1,Icell(i,2)+it2,Icell(i,3)+it3,l)
*Wgts(it3,it2,it1,i)
                end do
             end do
          end do
       end do
    end do

  END SUBROUTINE GAM_interpolate

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  FORCES
  SUBROUTINE GAM_computeFsr(X,q,Fsr,pnF,ptF)
    real, dimension(Natm,3) :: X
    real, dimension(Natm) :: Fsr,pnF,ptF
    real, dimension(Natm)   :: q
    real    :: rij1,rij2,fc,pnc,ptc
    real    :: fac1,fac2,r2,src2
    real, dimension(3)    :: xx,ff
    integer :: i,j,k,it

    Fsr = 0.
    pnF = 0.
    ptF = 0.

    src2 = srcutoff*srcutoff
    fac1 = REAL(Ntable)/(rc_sr**2*listfac**2)
    fac2 = 1./fac1

    do i = 1,Natm
        do k = nlist(i,1),nlist(i,2)
         j = list(k)%nbr
            xx(:) = X(i,:) - X(j,:)
            xx(:) = xx(:) - Lb(:)*nint(xx(:)/Lb(:))
            rij2 = (xx(1)*xx(1) + xx(2)*xx(2) + xx(3)*xx(3))
!          write(*,*) "i,j,rij2",i,j,rij2
          if (rij2 .lt. src2) then
           it = INT(rij2*fac1)
           r2 = REAL(it)*fac2
                pnc = q(i)*q(j)* ((Fsrtable(it+1)-Fsrtable(it))
*fac1*(rij2-r2) + Fsrtable(it))*2*xx(3)*xx(3)
                ptc =  q(i)*q(j)* ((Fsrtable(it+1)-Fsrtable(it))
*fac1*(rij2-r2) + Fsrtable(it))*(xx(1)*xx(1)+xx(2)*xx(2))
                fc = ptc - pnc
          Fsr(i) = Fsr(i) + 0.5*fc
                Fsr(j) = Fsr(j) + 0.5*fc
                pnF(i) = pnF(i) + 0.5*pnc
                ptF(i) = ptF(i) + 0.5*ptc
                pnF(j) = pnF(j) + 0.5*pnc
                ptF(j) = ptF(j) + 0.5*ptc
          end if
       end do
    end do

  END SUBROUTINE GAM_computeFsr

  SUBROUTINE GAM_makeFsrtable

    integer    :: i
    real    :: r2,a2,r6,gp,ar2

    a2 = alpha*alpha
    do i = 1,Ntable
       r2 = REAL(i)/REAL(Ntable)*rc_sr*rc_sr*listfac*listfac
       r6 = r2*r2*r2
       ar2 = a2*r2
       gp = EXP(-ar2)*(1+ar2+ar2*ar2/2.0+ar2*ar2*ar2/6.0)
       Fsrtable(i) =  (12*gp/r6/r2-24.0/r6/r6/r2)
    end do

  END SUBROUTINE GAM_makeFsrtable

  SUBROUTINE GAM_nlistFsr(X)
    real   ,dimension(Natm,3)                     :: X
! particle positions
    integer,dimension(Natm)                       :: LL
! next in list pointer
    integer,dimension(0:Ncsr(1)+1,0:Ncsr(2)+1,0:Ncsr(3)+1) ::
HOC ! H-of-C pointer with perioidic continuation
    real,dimension(3)      :: xx
    integer                :: ic1,ic2,ic3
    real                   :: rij2,fac2
    integer                :: i,j,n,l
    integer                :: nbr_cnt
    integer, dimension(3)  :: ndx

    integer,dimension(3*13), parameter :: poff =(/ 1 , 0 , 0 , &
                                                   1 , 0 , 1 , &
                                                   1 , 0 ,-1 , &
                                                   1 , 1 , 0 , &
                                                   1 , 1 , 1 , &
                                                   1 , 1 ,-1 , &
                                                   1 ,-1 , 0 , &
                                                   1 ,-1 , 1 , &
                                                   1 ,-1 ,-1 , &
                                                   0 , 1 , 1 , &
                                                   0 , 1 , 0 , &
                                                   0 , 1 ,-1 , &
                                                   0 , 0 , 1  /)
    call chainlist(LL,HOC,Ncsr,X)
```

```
10  continue

    nbr_cnt = 1
    fac2 = rc_sr*rc_sr*listfac*listfac
    do ic3 = 1,Ncsr(3)
       do ic2 = 1,Ncsr(2)
          do ic1 = 1,Ncsr(1)
             i = HOC(ic1,ic2,ic3) ! start with atom at the head
of the linked list
             do while(i.ne.0)   ! loop over all atoms in the
linked list if any
                nlist(i,1) = nbr_cnt

                ! IN CELL
                j = LL(i)      ! start with the next atom in the
linked list if any

                do while(j.gt.0)

                   xx = X(i,:) - X(j,:)
                   rij2 = xx(1)*xx(1)+xx(2)*xx(2)+xx(3)*xx(3)
                   if (rij2.lt.fac2) then

                      list(nbr_cnt)%offset(:) = 0
                      list(nbr_cnt)%nbr = j    ! neighbor index
                      nbr_cnt = nbr_cnt + 1
                      if (nbr_cnt.gt.MaxNbr) then
                         goto 20
                      endif

                   end if

                   j = LL(j)

                end do

                do n = 1,39,3  ! loop over 13 other cells
                 j = HOC(ic1+poff(n),ic2+poff(n+1),ic3+poff(n+2))
                  do while(j.ne.0)
                      xx = X(i,:) - X(j,:)
                      ndx = 0
                      do l = 1,3
                         if (xx(l).gt.Lb(l)/2.) then
                            xx(l) = xx(l) - Lb(l)
                            ndx(l) = -1
                         else if (xx(l).lt.-Lb(l)/2.) then
                            xx(l) = xx(l) + Lb(l)
                            ndx(l) = +1
                         end if
                      end do
                      rij2 = xx(1)*xx(1)+xx(2)*xx(2)+xx(3)*xx(3)
! rij2 must be less than the cutoff
                      if (rij2.lt.fac2) then
                         list(nbr_cnt)%offset(:) = ndx
! periodic offset factor
                         list(nbr_cnt)%nbr = j ! neighbor index
                         nbr_cnt = nbr_cnt + 1
                      if (nbr_cnt.gt.MaxNbr) then
                         goto 20
                      endif

                      end if

                      j = LL(j)

                   end do

                end do

                nlist(i,2) = nbr_cnt - 1
                i = LL(i)
             end do

          end do
       end do
    end do
20  continue
    if (nbr_cnt.gt.MaxNbr) then
       write(*,*) "nbr_cnt,MaxNbr",nbr_cnt,MaxNbr
       deallocate(list)
       MaxNbr = INT(1.5*nbr_cnt) + 1
       allocate(list(MaxNbr))
       goto 10
!!$       stop 'MaxNbr too small,nlist subroutine'
    else if (MaxNbr.gt.1.35*nbr_cnt.and.MaxNbr.gt.100) then
       deallocate(list)
       MaxNbr = INT(1.25*nbr_cnt)+1
       allocate(list(MaxNbr))
       write(out_unit,*)"Lowering MaxNbr to ",MaxNbr
       goto 10
    else if (MaxNbr.lt.1.05*nbr_cnt.and.MaxNbr.gt.100) then
       deallocate(list)
       MaxNbr = INT(1.25*nbr_cnt)+1
       allocate(list(MaxNbr))
       write(out_unit,*)"Raising MaxNbr to ",MaxNbr
       goto 10
    end if

!!$    print *,"Fsr nbr_cnt",nbr_cnt

  END SUBROUTINE GAM_nlistFsr

  SUBROUTINE GAM_computeR(X,q,gam,pnR,ptR)
    real,dimension(Natm,3)    :: X        ! particle positions
    real,dimension(Natm)      :: q        ! atomic charges
    real,dimension(Natm,3)    :: R        ! PM force (xyz)
    real,dimension(Natm)      :: gam,pnR,ptR ! Surface Tension
    integer,dimension(Natm,3) :: Icell  ! atom cell list
    real,dimension(Ncc(1),Ncc(2),Ncc(3))    :: rho
! mesh charge density
    real,dimension(Ncc(1),Ncc(2),Ncc(3),3)  :: F
! mesh force (xyz)
    real,dimension(-1:1,-1:1,-1:1,Natm)  :: Wgts
! interpolation/distribution weights
    integer                   :: i

    call atomcell(X,Icell,HH,Ncc)
    call GAM_weights(X,Icell,Wgts)
    call GAM_distribute(Icell,q,Wgts,rho)

    call GAM_poisson(F,rho)

    call GAM_interpolate(Icell,F,Wgts,R,3)
```

```
! multiply by charge to compute force at each atom
    do i = 1,Natm
       R(i,:) = R(i,:)*q(i)
!       write(*,*) R(i,1),R(i,2),R(i,3)
    end do
!       write(*,*)
    gam(:) = (R(:,1)+R(:,2)-2*R(:,3))/4.0
    pnR(:) = R(:,3)/2.0
    ptR(:) =  (R(:,1)+R(:,2))/4.0

  END SUBROUTINE GAM_computeR

  SUBROUTINE GAM_poisson(F,rho)
    real,dimension(Ncc(1),Ncc(2),Ncc(3),3) :: F
! force (xyz)
    real,dimension(Ncc(1),Ncc(2),Ncc(3))   :: rho
! mesh charge density

    complex,dimension(Ncc(1)/2+1,Ncc(2),Ncc(3))    :: rhohat
! transformed source
    complex,dimension(Ncc(1)/2+1,Ncc(2),Ncc(3),3)  :: fhat
! transformed force

    integer          ::  k1,k2,k3

! Solve "Poisson" -- multiply by optimal influence function
    call rfftwnd_f77_one_real_to_complex(plan_r2c,rho,rhohat)
    rhohat = rhohat*Ghat

! Compute force (multiply by wave number i kx,i ky,i kz)
    do k3 = 1,Ncc(3)
       do k2 = 1,Ncc(2)
          do k1 = 1,Ncc(1)/2+1
             fhat(k1,k2,k3,1) = rhohat(k1,k2,k3)*(-1.,0.)*kx(k1)*
kx(k1)
             fhat(k1,k2,k3,2) = rhohat(k1,k2,k3)*(-1.,0.)*ky(k2)*
ky(k2)
             fhat(k1,k2,k3,3) = rhohat(k1,k2,k3)*(-1.,0.)*kz(k3)*
kz(k3)
          end do
```

```
         end do
      end do

! Inverse transform potential and forces
      call rfftwnd_f77_one_complex_to_real(plan_c2r,fhat(1,1,1,1)
,F(1,1,1,1))
      call rfftwnd_f77_one_complex_to_real(plan_c2r,fhat(1,1,1,2)
,F(1,1,1,2))
      call rfftwnd_f77_one_complex_to_real(plan_c2r,fhat(1,1,1,3)
,F(1,1,1,3))

   END SUBROUTINE GAM_poisson

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FFT stuff
   SUBROUTINE GAM_initPPPM

      allocate (Ghat(Ncc(1)/2+1,Ncc(2),Ncc(3)))
      allocate (list(MaxNbr),nlist(Natm,2))
      allocate (Fsrtable(Ntable))

      rc_sr = srcutoff*sigmax
      Ncsr = MAX(1,INT(Lb/rc_sr))
      print *,"Ncsr = ",Ncsr

   END SUBROUTINE GAM_initPPPM

   SUBROUTINE GAM_newalpha

      call GAM_computeGhat
      call GAM_makeFsrtable

   END SUBROUTINE GAM_newalpha

   SUBROUTINE GAM_computeGhat

      integer    :: k1,k2,k3,kk2,kk3

      do k1 = 0,Ncc(1)/2
         do k2 = 0,Ncc(2)/2
            do k3 = 0,Ncc(3)/2
           if (k1.ne.0 .or. k2.ne.0 .or. k3.ne.0) call GAM_GetGhat
(k1,k2,k3,Ghat(k1+1,k2+1,k3+1))
            end do
            do k3 = Ncc(3)/2+1,Ncc(3)-1
               kk3 = Ncc(3)-k3
          if (k1.ne.0 .or. k2.ne.0 .or. kk3.ne.0) call GAM_GetGhat
(k1,k2,kk3,Ghat(k1+1,k2+1,k3+1))
            end do
         end do
         do k2 = Ncc(2)/2+1,Ncc(2)-1
            kk2 = Ncc(2)-k2
            do k3 = 0,Ncc(3)/2
          if (k1.ne.0 .or. kk2.ne.0 .or. k3.ne.0) call GAM_GetGhat
(k1,kk2,k3,Ghat(k1+1,k2+1,k3+1))
            end do
            do k3 = Ncc(3)/2+1,Ncc(3)-1
               kk3 = Ncc(3)-k3
         if (k1.ne.0 .or. kk2.ne.0 .or. kk3.ne.0) call GAM_GetGhat
(k1,kk2,kk3,Ghat(k1+1,k2+1,k3+1))
            end do
         end do
      end do
      Ghat(1,1,1) = 0.
!    Ghat = -Ghat/(4.*Pi)/eps0/(Lb(1)*Lb(2)*Lb(3))
      Ghat = -Ghat/(Lb(1)*Lb(2)*Lb(3))

   END SUBROUTINE GAM_computeGhat

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FFT stuff
   SUBROUTINE GAM_initfft

      integer    :: k1,k2,k3

      allocate (kx(Ncc(1)/2+1),ky(Ncc(2)),kz(Ncc(3)))

      call rfftw3d_f77_create_plan(plan_c2r,Ncc(1),Ncc(2),Ncc(3),
+1,0) !1)  ! comp_to_real
      call rfftw3d_f77_create_plan(plan_r2c,Ncc(1),Ncc(2),Ncc(3),
-1,0) !1)  ! real_to_comp
```

```
!  Compute wave numbers for xyz derivatives
    do k1 = 0,Ncc(1)/2
       kx(k1+1) = REAL(k1)*2.*Pi/Lb(1)
    end do
    do k2 = 0,Ncc(2)/2
       ky(k2+1) = REAL(k2)*2.*Pi/Lb(2)
    end do
    do k2 = Ncc(2)/2+1,Ncc(2)-1
       ky(k2+1) = -REAL(Ncc(2)-k2)*2.*Pi/Lb(2)
    end do
    do k3 = 0,Ncc(3)/2
       kz(k3+1) = REAL(k3)*2.*Pi/Lb(3)
    end do
    do k3 = Ncc(3)/2+1,Ncc(3)-1
       kz(k3+1) = -REAL(Ncc(3)-k3)*2.*Pi/Lb(3)
    end do

  END SUBROUTINE GAM_initfft


  SUBROUTINE GAM_getU2sum(k1,k2,k3,U2)
    integer  :: k1,k2,k3
    real  :: rk1,rk2,rk3,U2

    rk1 = REAL(k1)*2.*Pi/Lb(1)
    rk2 = REAL(k2)*2.*Pi/Lb(2)
    rk3 = REAL(k3)*2.*Pi/Lb(3)

    U2 =   (1. - SIN(rk1*HH(1)/2.)**2 + 2./15.*SIN(rk1*HH(1)/
2.)**4) &
         * (1. - SIN(rk2*HH(2)/2.)**2 + 2./15.*SIN(rk2*HH(2)/
2.)**4) &
         * (1. - SIN(rk3*HH(3)/2.)**2 + 2./15.*SIN(rk3*HH(3)/
2.)**4)

  END SUBROUTINE GAM_getU2sum


  SUBROUTINE GAM_getRU2sum(k1,k2,k3,RU2)
    integer  :: k1,k2,k3
    integer  :: m1,m2,m3
```

```
real  :: rk1,rk2,rk3
real  :: rm1,rm2,rm3
real, dimension(3)  :: R,RU2
real                :: a,b,c,f1,f2,f3
integer, parameter :: Mmax=4

rk1 = REAL(k1)*2.*Pi/Lb(1)
rk2 = REAL(k2)*2.*Pi/Lb(2)
rk3 = REAL(k3)*2.*Pi/Lb(3)

RU2 =0.
do m1 = -Mmax,Mmax

   rm1 = REAL(m1)
   a = rk1+2.*Pi*rm1/HH(1)
   if (a.eq.0) then
      f1 = 1.
   else
      f1 = SIN(a*HH(1)/2.)/(a*HH(1)/2.)
   end if

   do m2 = -Mmax,Mmax

      rm2 = REAL(m2)
      b = rk2+2.*Pi*rm2/HH(2)
      if (b.eq.0) then
         f2 = 1.
      else
         f2 = SIN(b*HH(2)/2.)/(b*HH(2)/2.)
      end if

      do m3 = -Mmax,Mmax

         rm3 = REAL(m3)
         c = rk3+2.*Pi*rm3/HH(3)
         if (c.eq.0) then
            f3 = 1.
         else
            f3 = SIN(c*HH(3)/2.)/(c*HH(3)/2.)
         end if
```

```
          call GAM_getR(a,b,c,R)
          RU2 = RU2  + R*(f1*f2*f3)**6

       end do
    end do
 end do

END SUBROUTINE GAM_getRU2sum

SUBROUTINE GAM_getGhat(k1,k2,k3,Ghat)
 integer  :: k1,k2,k3
 real  :: Ghat,U2
 real, dimension(3)  :: RU2,D

 call GAM_getRU2sum(k1,k2,k3,RU2)
 call GAM_getU2sum(k1,k2,k3,U2)
 call GAM_getD(k1,k2,k3,D)

 Ghat = (D(1)*RU2(1) + D(2)*RU2(2) + D(3)*RU2(3) )/ &
      ( (D(1)*D(1) + D(2)*D(2) + D(3)*D(3))*U2*U2 )

END SUBROUTINE GAM_getGhat

SUBROUTINE GAM_getD(k1,k2,k3,D)
 integer  :: k1,k2,k3
 real  :: rk1,rk2,rk3
 real,dimension(3) :: D

 rk1 = REAL(k1)*2.*Pi/Lb(1)
 rk2 = REAL(k2)*2.*Pi/Lb(2)
 rk3 = REAL(k3)*2.*Pi/Lb(3)

 D(1) = rk1 ; D(2) = rk2 ; D(3) = rk3

END SUBROUTINE GAM_getD


SUBROUTINE GAM_getR(rk1,rk2,rk3,R)
 real  :: rk1,rk2,rk3
 real,dimension(3) :: R
```

```
    real  :: ksq,f,kp,Yc,f1,f2,a2,k1

    real  :: DERF

    ksq = (rk1*rk1 + rk2*rk2 + rk3*rk3)
    k1 = SQRT(ksq)
    kp = SQRT(rk1*rk1+rk3*rk3)
    Yc = Lb(2)*.5
    a2 = alpha*alpha

    if (ksq.gt.0) then
f1 = 2*Pi*srPi*alpha*(exp(-ksq/4./a2) - k1*srPi/2./
alpha*(1-ERF(k1/2./alpha)))
f2 = 1.
       R(1) = -rk1*f1*f2  ! leaving out i
       R(2) = -rk2*f1*f2  ! leaving out i
       R(3) = -rk3*f1*f2  ! leaving out i

    else

       R = 1.

    end if

  END SUBROUTINE GAM_getR

  SUBROUTINE GAM_endfft

     call rfftwnd_f77_destroy_plan(plan_r2c)
     call rfftwnd_f77_destroy_plan(plan_c2r)

   END SUBROUTINE GAM_endfft

END MODULE p3mgam
!!!!
!! This is uni.f
!! to generate the random numbers
!!!!
      REAL FUNCTION UNI(JD)
C***BEGIN PROLOGUE  UNI
C***DATE WRITTEN   810915
```

```
C***REVISION DATE  830805
C***CATEGORY NO.  L6A21
C***KEYWORDS  RANDOM NUMBERS, UNIFORM RANDOM
NUMBERS
C***AUTHOR    BLUE, JAMES, SCIENTIFIC COMPUTING
DIVISION, NBS
C   KAHANER, DAVID, SCIENTIFIC COMPUTING DIVISION, NBS
C   MARSAGLIA, GEORGE, COMPUTER SCIENCE DEPT., WASH STATE UNIV
C
C THIS ROUTINE GENERATES QUASI UNIFORM RANDOM NUMBERS ON [0,1
C AND CAN BE USED ON ANY COMPUTER WITH WHICH ALLOWS INTEGERS
C              AT LEAST AS LARGE AS 32767.
C***DESCRIPTION
C
C       THIS ROUTINE GENERATES QUASI UNIFORM RANDOM
C NUMBERS ON THE INTER
C       [0,1).  IT CAN BE USED WITH ANY COMPUTER WHICH
C       INTEGERS AT LEAST AS LARGE AS 32767.
C
C
C   USE
C       FIRST TIME....
C                   Z = UNI(JD)
C                     HERE JD IS ANY  N O N - Z E R O  INTEGER.
C                     THIS CAUSES INITIALIZATION OF THE PROGRAM
C AND THE FIRST RANDOM NUMBER TO BE RETURNED AS Z.
C       SUBSEQUENT TIMES...
C                   Z = UNI(0)
C       CAUSES THE NEXT RANDOM NUMBER TO BE RETURNED AS Z.
C
C
C.....................................................
C   NOTE: USERS WHO WISH TO TRANSPORT THIS PROGRAM FR
OM ONE COMPUTER
C         TO ANOTHER SHOULD READ THE FOLLOWING INFORMATI
ON.....
C
C   MACHINE DEPENDENCIES...
C      MDIG = A LOWER BOUND ON THE NUMBER OF BINARY DIGI
TS AVAILABLE
C          FOR REPRESENTING INTEGERS, INCLUDING THE SIGN BIT.
```

```
C          THIS VALUE MUST BE AT LEAST 16, BUT MAY BE INCREASED
C              IN LINE WITH REMARK A BELOW.
C
C   REMARKS...
C     A. THIS PROGRAM CAN BE USED IN TWO WAYS:
C      (1) TO OBTAIN REPEATABLE RESULTS ON DIFFERENT COMPUTERS,
C      SET 'MDIG' TO THE SMALLEST OF ITS VALUES ON EACH, OR,
C      (2) TO ALLOW THE LONGEST SEQUENCE OF RANDOM NUMBERS TO BE
C      GENERATED WITHOUT CYCLING (REPEATING) SET 'MDIG' TO THE
C      LARGEST POSSIBLE VALUE.
C     B. THE SEQUENCE OF NUMBERS GENERATED DEPENDS ON THE INITIAL
C          INPUT 'JD' AS WELL AS THE VALUE OF 'MDIG'.
C          IF MDIG=16 ONE SHOULD FIND THAT
C            THE FIRST EVALUATION
C              Z=UNI(305) GIVES Z=.027832881...
C            THE SECOND EVALUATION
C              Z=UNI(0) GIVES   Z=.56102176...
C            THE THIRD EVALUATION
C              Z=UNI(0) GIVES   Z=.41456343...
C            THE THOUSANDTH EVALUATION
C              Z=UNI(0) GIVES   Z=.19797357...
C
C***REFERENCES  MARSAGLIA G.,
C "COMMENTS ON THE PERFECT UNIFORM RANDOM
C  NUMBER GENERATOR", UNPUBLISHED NOTES, WAS H S. U.
C***ROUTINES CALLED  I1MACH,XERROR
C***END PROLOGUE  UNI
      INTEGER M(17)
C
      SAVE I,J,M,M1,M2
C
      DATA M(1),M(2),M(3),M(4),M(5),M(6),M(7),M(8),M(9),M(1
0),M(11),
     1     M(12),M(13),M(14),M(15),M(16),M(17)
     2              / 30788,23052,2053,19346,10646,19427,23975,
     3              19049,10949,19693,29746,26748,2796,23890,
     4              29168,31924,16499 /
      DATA M1,M2,I,J / 32767,256,5,17 /
C***FIRST EXECUTABLE STATEMENT  UNI
      IF(JD .EQ. 0) GO TO 3
C  FILL
```

```
C          BE SURE THAT MDIG AT LEAST 16...
      MDIG = 32
      M1= 2**(MDIG-2) + (2**(MDIG-2)-1)
      M2 = 2**(MDIG/2)
      JSEED = MIN0(IABS(JD),M1)
      IF( MOD(JSEED,2).EQ.0 ) JSEED=JSEED-1
      K0 =MOD(9069,M2)
      K1 = 9069/M2
      J0 = MOD(JSEED,M2)
      J1 = JSEED/M2
      DO 2 I=1,17
        JSEED = J0*K0
        J1 = MOD(JSEED/M2+J0*K1+J1*K0,M2/2)
        J0 = MOD(JSEED,M2)
    2   M(I) = J0+M2*J1
      I=5
      J=17
C  BEGIN MAIN LOOP HERE
    3 K=M(I)-M(J)
      IF(K .LT. 0) K=K+M1
      M(J)=K
      I=I-1
      IF(I .EQ. 0) I=17
      J=J-1
      IF(J .EQ. 0) J=17
      UNI=FLOAT(K)/FLOAT(M1)
      RETURN
      END
!!!!
!! This is input.dat
!! input file for the P3M simulation
!!!!
1000           Natm
1              Ntyp

32             Nc(1)
32             Nc(2)
64             Nc(3)
48             Ncc(1)
48             Ncc(2)
96             Ncc(3)
```

```
1100000        Nt
0.005          Ts

12             Lb - x
12             Lb - y
24             Lb - z

0.85           Ttar
50000          Ts1
100000         Ts2
0.05           tau1
0.05           tau2
0              Twal
1.E-24         Q
1.13E5         Prs
1.E-24         Ms

-10         tmp_out
-10         eng_out
-100        max_out
-100        rdf_out
-100        btz_out
100         atm_out
5000        ubr_out
5000        res_out
-1          mfx_out

100           Nmaxg
30            Nmaxw
0.            Vmax_peak
512           Nbin
2.5           ljcutoff  (r/sig)
3.0           srcutoff  (r/sig)
1.15          listfac
10            ntlist
10000         Ntable
4000000       MaxNbr
4000000       MaxNbrsr
1.0           alpha

0           I_time
```

```
-1          I_whichic
1.0         Alj
1.0         Clj
1.0         sigmax
!!!!
!! This is makefile
!! make the excutable code using FFTW library
!!!!
#FORT    =  f90
#FORT77  =  g77
#FORT    = fort
#FORT77  = fort
FORT    =  lf95
FORT77  =  lf95
NICE    = --dbl
#NICE    = -fpconstant -real_size 64
LIB     = -lrfftw -lfftw
#LIB     = -lrfftw -lfftw -non_shared
OPTS    =  -O
#OPTS    =  -O5 -tune ev6
#OPTS    =  -fpconstant -real_size 64
#PROF   = -pg
#DEBUG   =  -g --chk

OBJECTS = prms.o data.o main.o nlistmod.o uni.o p3mlj.o p3mgam.o

run:  $(OBJECTS) makefile
$(FORT) $(LIB) -o run $(PROF) $(OBJECTS) $(LIB)

main.o:   main.f90 prms.o data.o nlistmod.o p3mlj.o p3mgam.o
makefile
$(FORT) -c $(NICE) $(OPTS) $(PROF) $(DEBUG)  main.f90

data.o:  data.f90 prms.o makefile
$(FORT) -c $(NICE) $(OPTS) $(PROF) $(DEBUG)  data.f90

nlistmod.o:  nlistmod.f90 prms.o data.o makefile
$(FORT) -c $(NICE) $(OPTS) $(PROF) $(DEBUG) nlist
mod.f90

p3mlj.o:  p3mlj.f90 prms.o nlistmod.o  makefile
```

```
$(FORT) -c $(NICE) $(OPTS) $(PROF) $(DEBUG)  p3mlj.f90

p3mgam.o:  p3mgam.f90 prms.o nlistmod.o  makefile
$(FORT) -c $(NICE) $(OPTS) $(PROF) $(DEBUG)  p3mgam.f90

prms.o:  prms.f90 makefile
$(FORT) -c $(NICE) $(OPTS) $(PROF) $(DEBUG)  prms.f90

uni.o:  uni.f makefile
$(FORT77) -c $(NICE) uni.f
```

Appendix B: Code for Droplet Simulation

```
ccccccccccccccccccccccccccccccccccccccccccccccc
c   This code uses Verlet neighborlist method for liquid
c   droplet simulation (Chapter 4) on the FCC solid
c   surface.
ccccccccccccccccccccccccccccccccccccccccccccccc
c   Simulation of thin liquid film adjacent to solid
c   surface uses the same code, but sample the
c   density and surface tension differently. In case
c   of thin film, density is sampled only as function
c   of ‘‘z’’ and surface tension is also sampled.
ccccccccccccccccccccccccccccccccccccccccccccccc
c
c   This is adjustcom.f
c   to adjust the center of mass of the system
c
      subroutine adjustcom
      include "head.h"
      sum = 0
      sumvx = 0
      sumvy = 0
      do i = 1,nmol
         sumvx = sumvx + x(i)
         sumvy = sumvy + y(i)
      enddo
      do i = 1,nmol
         x(i) = x(i) - sumvx/nmol
         y(i) = y(i) - sumvy/nmol
         if (x(i).gt.box/2) then
            x(i) = x(i) - box
         else if (x(i).lt.(-box/2)) then
            x(i) = x(i) + box
         end if
         if (y(i).gt.boxy/2) then
            y(i) = y(i) - boxy
         else if (y(i).lt.(-boxy/2)) then
            y(i) = y(i) + boxy
         end if
      enddo
c     write(6,*)'sum_fin=',sum/(3*nmol)
      return
      end
c
c  This is adjust.f
c  to adjust the system temperature using berendsen thermostat
c
      subroutine adjust
      include "head.h"
      sum = 0
      sumvx = 0
      sumvy = 0
      sumvz = 0
      do i = 1,nmol
         sumvx = sumvx + x1(i)
         sumvy = sumvy + y1(i)
         sumvz = sumvz + z1(i)
      enddo
      do i = 1,nmol
         x1(i) = x1(i) - sumvx/nmol
         y1(i) = y1(i) - sumvy/nmol
         z1(i) = z1(i) - sumvz/nmol
         sum = sum + x1(i)**2+y1(i)**2+z1(i)**2
      enddo
c     write(6,*)'sum_ear=',sum/(3*nmol)
c     fs = sqrt(3*temp*nmol/sum)
c     berendsen's thermostat
      sum = sum/(3*nmol)
      fs = sqrt(1 + (temp/sum - 1)/20.0)
      do i = 1,nmol
         x1(i) = x1(i)*fs
         y1(i) = y1(i)*fs
         z1(i) = z1(i)*fs
      enddo
      sum = 0
      do i = 1,nmol
         sum = sum + x1(i)**2+y1(i)**2+z1(i)**2
      enddo
c     write(6,*)'sum_fin=',sum/(3*nmol)
      return
      end
c
c  This is create_nbr.f
c  to create the neighbor list
c
      subroutine create_nbr
      include "head.h"
      k = 0
      do i = 1,nmol
         ipoint(i) = k + 1
         do j = i+1,nmol
            xr = x(i) - x(j)
            xr = xr - box*nint(xr/box)
            yr = y(i) - y(j)
            yr = yr - boxy*nint(yr/boxy)
            zr = z(i) - z(j)
            rij = sqrt(xr**2+yr**2+zr**2)
            if (rij.lt.rl) then
               k = k + 1
c              write(*,*) k
               list(k) = j
            end if
         enddo
      enddo
      return
      end
c
c   This is findtemp.f
c   to find the temperature of the system
c
      subroutine findtemp(temperature)
      include "head.h"
      sum = 0
      do i = 1,nmol
         sum = sum + x1(i)**2+y1(i)**2+z1(i)**2
      enddo
      temperature = sum/(nmol*3)
      return
      end
c
c  This is force.f
```

```
c  to calculate the forces on atoms
c
      subroutine force
      include "head.h"
c     write(*,*) 'entering force'
      rc6i = 1/(rc**6)
      rc2 = rc**2
c     ecut = 4*rc6i*(rc6i-1)
      do i = 1,nmol
         fx(i) = 0
         fy(i) = 0
         fz(i) = 0
      enddo
      do i = 1,nmol-1
         do k = ipoint(i),ipoint(i+1) - 1
            xr = x(i)-x(list(k))
            yr = y(i)-y(list(k))
            zr = z(i)-z(list(k))
            xr = xr - box*nint(xr/box)
            yr = yr - boxy*nint(yr/boxy)
c           zr = zr - box*nint(zr/boxz)
            r2 = xr**2+yr**2+zr**2
            if (r2.lt.rc2) then
               r2i = 1/r2
               r6i = r2i**3
               ff = 48*r2i*r6i*(r6i-0.5)
               fx(i) = fx(i) + ff*xr
               fy(i) = fy(i) + ff*yr
               fz(i) = fz(i) + ff*zr
               fx(list(k)) = fx(list(k)) - ff*xr
               fy(list(k)) = fy(list(k)) - ff*yr
               fz(list(k)) = fz(list(k)) - ff*zr
c              en = 4*r6i*(r6i-1) - ecut
            end if
         enddo
      enddo
c
      pi = 4*atan(1.0)
      sig3 = sig_int**3
      sig6 = sig3*sig3
      constant = 4.0*pi*epsi_int*sig6/(5.0*sqrt(3.0)*a1*a1*a1)
      do i = 1,nmol
       zr = boxz/2.0 + z(i)
       fz(i) = fz(i) + constant*(1.0/zr)**4*(2*(sig_int/zr)**6-5)
      enddo
      do i = 1,nmol
         x2(i) = fx(i)
         y2(i) = fy(i)
         z2(i) = fz(i)
      enddo
c     write(6,*) 'leaving force'
      return
      end
c
c  This is input.f
c  to read input data from "input.dat" file
c
       subroutine input
**
c      real mh,mo,l_com
       include "head.h"
**
       write(6,*) 'entering input'
       open(unit=10,file = 'input.dat')
          read(10,*) nmol,max_cycle,nsamp,nhis,nstep
          read(10,*) temp,h,rc,rl,istart,sig_int,epsi_int,a1
          read(10,*) box,boxy,boxz
       close(10)
*
       write(6,*) 'leaving input'
c---------------------------------------------------------------
       return
       end
c
c   This is main.f
c   Main program section
c
      program main
C     Main Program
c     This program calls different subroutines for force evaluation,
c     motion according to newton's law, sampling of various properties.
      include "head.h"
```

```
c     Now it is time to read the input, supply input data.
      call input
      rvl = rl - rc
c     rl : nieghbor list radii ; rc: cutoff radii
c     rvl : if any molecule moves by the distance rvl,
c     then it will
c     become important to check if we need to update the
c     neighbor list
      a2 = a1*sqrt(2.0/3.0)
c     not reuqired : probably
      write(6,*)  box, boxy, boxz
c     dimension of the simulation box
C     Input file read
      nrun = 0
c     nrun counts the number of time we sampled any properties
      sur_tension = 0
      xx = 0
c     intializing variables and array
      nstart = 150000
c     After nstart time steps, we will start sampling
      do i = 1,nhis+1
         do j = 1,nhis+1
            do k = 1,nhis+1
                igz(i,j,k) = 0
            enddo
          enddo
      enddo

c
c     This variables are defined in the subroutine gamma.f
c     gz : density distribution as z
c

c     intializing velocity distribution, to compare with
c     maxwell boltzman
c     -man  distribution
c
      if (istart.eq.1) then
        call restart
c     read position etc. from existing result files
        call findtemp(xx)
```

```
c     find the temperature of the system
      else
        call start
c     initialize some variables, check start.f file
        call findtemp(xx)
        istart_step=0
      end if
      call output(istart_step)
c     write initial result files
c     delv : bin for the velocity distribution
      delx = box/2.0/nhis
      dely = boxy/2.0/nhis
      delz = boxz/2.0/nhis
      rho = nmol/(box*boxy*boxz)
      write(*,*) 'delx,dely,delz =',delx,dely,delz
c     Non dimesionalized density of the system (overall, not
c     local)
      call create_nbr
c     create neighbor list, used in the neighbor list algorithm
      do ii = 1,nmol
         xold(ii) = x(ii)
         yold(ii) = y(ii)
         zold(ii) = z(ii)
      enddo
c     refernce position when neighbor list is first created
c     surface tension (temporary), not surface temperature
      open(unit=15, file="temp.dat")
      call force
      do  i = istart_step + 1,max_cycle
c        write(*,*)'i=',i
         call verlet
         call testnbrlist(i)
         if (i.le.100000) then
            call adjust
         else if (mod(i,5).eq.0) then
            call adjust
         end if
         if (mod(i,nsamp).eq.0) then
            call findtemp(xx)
            write(15,*) i , xx
            write(*,*) i , xx
            if (mod(i,500).eq.0) then
               call adjustcom
               call create_nbr
               do ii= 1,nmol
                  xold(ii) = x(ii)
                  yold(ii) = y(ii)
                  zold(ii) = z(ii)
               enddo
             endif
         endif
c
         if ((mod(i,nsamp).eq.0).and.(i.ge.nstart)) then
            nrun = nrun + 1
            call sample
c           write(6,*) 'nrun =',nrun
         end if
         open (unit=11, file = 'check')
            read(11,*) icheck
         close(11)
         if (icheck.eq.1) then
            call output(i)
            go to 1000
         end if
         if (mod(i,nstep).eq.0) then
            call output(i)
         end if
      enddo
      close(15)

c end of the simulation: post processing next
c file units are 11: check , 12: gr.dat, 13:res....dat ,14:gv.dat

      call output(max_cycle)
      write(6,*) 'leaving main'
1000  stop
      end
c
c  This is output.f
c  to write out the data at desired interval
c
      subroutine output(icycle)
```

```
*     subroutine writes the outfut file
      include "head.h"
      character(15) name, name1
c     write(6,*) 'entering output'
*
      itemp = icycle/nstep
      i100 = itemp/100
      i1 = itemp - i100*100
      i10 = i1/10
      i1 = i1 - i10*10
      name = 'res'//char(i100+48)//char(i10+48)//char(i1+48)//'
     1 .dat'
      name1 = 'den'//char(i100+48)//char(i10+48)//char(i1+48)//'
     1 .dat'
      open(unit=13, file="result/"//name)

c     file will be "resXXX.dat"
      write(13,*) nmol,nhis,h,temp,icycle,nsamp,max_cycle,rc
      write(13,*) box,boxy,boxz
      write(13,*) nrun
      do i = 1,nmol
         write(13,101) x(i),y(i),z(i),x1(i),y1(i),z1(i)
      enddo
      if (mod(icycle,400000).eq.0) then
         open(unit=14, file="result/"//name1)
         do i = 1,nhis
            do j=1,nhis
               do k =1,nhis
                  write(14,*) i,j,k,igz(i,j,k)
               enddo
            enddo
         enddo
         close(14)
      endif
      close(13)
 101  format(6E25.15)
 102  format(2I7,E25.15)
 103  format(E25.15)
c     write(6,*) 'leaving output'
      return
      end
```

```
c
c  This is random.f
c  to generate random numbers
c
      subroutine random(seed,randx)
*     subroutine generates random number from integer seed
        integer seed
        double precision randx
        seed = 2045*seed +1
        seed = seed - (seed/1048576)*1048576
        randx = real(seed + 1)/1048577.0
      return
      end
c
c  This is restart.f
c  to restart the incomplet program
c
      subroutine restart
c
c     subroutine reads  r, q & w values from the existing file .
c
      include "head.h"
c
      open(unit=20, file= 'res.dat')
      open(unit=21, file= 'den.dat')
      read(20,*) nmol,nhis,h,temp,istart_step,nsamp,max_1,rc
      read(20,*) box,boxy,boxz
      read(20,*) nrun
      do i = 1,nmol
         read(20,*) x(i),y(i),z(i),x1(i),y1(i),z1(i)
      enddo
      do i = 1,nhis
         do j =1,nhis
         do k = 1,nhis
         read(20,*) ii,jj,kk,igz(i,j,k)
         enddo
         enddo
      enddo
      close(20)
      write(*,*) 'nmol=',nmol
      write(*,*) 'max_cycle=',max_cycle
      write(*,*) 'start_step=',istart_step
      write(*,*) 'box=',box,boxy,boxz
      return
      end
c
c   This is sample.f
c   to sample density distribution
c
      subroutine sample
c
      include "head.h"
c     write(6,*) 'entering sample'
c     sampling densities
       do i = 1,nmol
          if ((x(i).ge.0).and.(y(i).ge.0).and.(z(i).le.0)) then
             xr = x(i)
             yr = y(i)
             zr = z(i) + boxz/2.0
             ibin1 = int(xr/delx)+1
             ibin2 = int(yr/dely)+1
             ibin3 = int(zr/delz) +1
             igz(ibin1,ibin2,ibin3) = igz(ibin1,ibin2,ibin3) + 1
          endif
      enddo
c     write(6,*) 'leaving sample'
      return
      end
c
c   This is start.f
c   to start the program from initial condition
c
      subroutine start
c     subroutine generates  r and its derivative

c     implicit double precision(a-h,o-z)

      include "head.h"
      double precision ran
      integer seed
      write(6,*)'entering start'
      seed = 12345
      sumvx = 0
      sumvy = 0
      sumvz = 0
      sumv2 = 0
      dx = 1.13
      dy = 1.13
      dz = 1.13
      nx = int(nmol**(1.0/3.0))
      ny = nx
      nxny = nx*ny
write(*,*) nx,ny,nxny
c     Initializing positions of liquid and vapor atoms
      do i = 1,nmol
         k1 = (i-1)/nxny
         j1 = (i - nxny*k1 - 1)/nx
         i1 = i - nxny*k1 -nx*j1
         j1 = j1 + 1
         k1 = k1 + 1
         x(i) = (i1-1)*dx - nx*dx/2.0
         y(i) = (j1-1)*dy - ny*dy/2.0
         z(i) = -boxz/2 + sig_int + (k1 - 1)*dz
      enddo
c
c     Initializing velocities of all the atoms.
c
      do i = 1,nmol
         call random(seed,ran)
         x1(i) = ran - .5
         call random(seed,ran)
         y1(i) = ran - .5
         call random(seed,ran)
         z1(i) = ran - .5
         sumvx = sumvx + x1(i)
         sumvy = sumvy + y1(i)
         sumvz = sumvz + z1(i)
      enddo
      do i = 1,nmol
         x1(i) = x1(i) - sumvx/nmol
         y1(i) = y1(i) - sumvy/nmol
         z1(i) = z1(i) - sumvz/nmol
         v2 = x1(i)**2 + y1(i)**2 + z1(i)**2
         sumv2 = sumv2 + v2
      enddo
c   Sum(v_i) = 0, velocity^2 proportional to temperature
      fs = sqrt(3*temp*nmol/sumv2)
      do i = 1,nmol
         x1(i) = x1(i) * fs
         y1(i) = y1(i) * fs
         z1(i) = z1(i) * fs
      enddo
c
       write(6,*) 'leaving start'
       return
       end
c
c   This is testnbrlist.f
c   to check if new neighborlist needs to be created
c
       subroutine testnbrlist(index)
       include "head.h"
       iflag = 0
       do i = 1, nmol
          xr = x(i) - xold(i)
          xr = xr - box*nint(xr/box)
          yr = y(i) - yold(i)
          yr = yr - boxy*nint(yr/boxy)
          zr = z(i) - zold(i)
          delta_r = sqrt(xr**2+yr**2+zr**2)
          if (delta_r.ge.rvl) then
             call create_nbr
c            write(6,*) 'nbrlistcreated',index,'mol#',i
             iflag = 1
             goto 100
          end if
       enddo
 100   continue
       if (iflag.eq.1) then
          do i = 1,nmol
             xold(i) = x(i)
             yold(i) = y(i)
             zold(i) = z(i)
          enddo
```

```
        end if
        return
        end
c
c   This is verlet.f
c   to integrate the motion of atoms using verlet algorithm
c
      subroutine verlet
c     using Velocity Verlet algorithm to integrate equation of
c     motion
      include "head.h"
c     write(*,*) 'entering verlet'
c     write(6,*) x1(20),y1(20),z1(20),x2(20),y2(20),z2(20)
      do i = 1,nmol
         x(i) = x(i) + h*x1(i) + h*h*x2(i)/2.0
         y(i) = y(i) + h*y1(i) + h*h*y2(i)/2.0
         z(i) = z(i) + h*z1(i) + h*h*z2(i)/2.0
         x1(i) = x1(i) + h*x2(i)/2
         y1(i) = y1(i) + h*y2(i)/2
         z1(i) = z1(i) + h*z2(i)/2
         if (x(i).gt.box/2) then
            x(i) = x(i) - box
         else if (x(i).lt.(-box/2)) then
            x(i) = x(i) + box
         end if
         if (y(i).gt.boxy/2) then
            y(i) = y(i) - boxy
         else if (y(i).lt.(-boxy/2)) then
            y(i) = y(i) + boxy
         end if
         if (z(i).gt.boxz/2) then
            z(i) = boxz - z(i)
            z1(i) = -z1(i)
         else if (z(i).lt.(-boxz/2)) then
            z(i) = - z(i) - boxz
            z1(i) = -z1(i)
         end if
c        x(i) = mod(x(i),box/2)
c        y(i) = mod(y(i),boxy/2)
c        z(i) = mod(z(i),boxz/2)
      enddo
      call force
      do i = 1,nmol
         x1(i) = x1(i) + h*x2(i)/2
         y1(i) = y1(i) + h*y2(i)/2
         z1(i) = z1(i) + h*z2(i)/2
      enddo
c     write(6,*) 'leaving  verlet'
      return
      end
```

References

[1] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen. A smooth particle mesh ewald method. *Journal of Chemical Physics*, 103(19):8577–8593, 1995.

[2] D. E. Sullivan. Surface tension and contact angle of liquid-solid interface. *Journal of Chemical Physics*, 74(4):2604–2615, 1981.

[3] A. W. Adamson. Potential distortion model for contact angle and spreading ii. temperature dependent effects. *Journal of Collid and Interface Science*, 44(2):293–299, 1973.

[4] J. H. Lay and 1995 Dhir, V. K. A theoretical study of the relationship between contact angles and the shape of a vapor stem. *Conference Proc. ASME Heat Transfer Divison*, pages 133–143, 1995.

[5] R. H. Fowler. *Proc. of Royal Society*, A159:221, 1937.

[6] J. G. Kirkwood and F. P. Buff. The statistical mechanical theory of surface tension. *Journal of Chemical Physics*, 17(3):338–343, 1949.

[7] A. Harasima. Molecular theory of surface tension. *Advances in Chemical Physics*, pages 203–237, 1950.

[8] R.C. Tolman. *Journal of Chemical Physics*, 17:333, 1949.

[9] C. Haye, M. J.; Bruin. A molecular dynamics study of the curvature correction to the surface tension. *Journal of Chemical Physics*, 100(1):556, 1994.

[10] M. J. P. Nijmeijer, C. Bruin, A. B. van Woerkom, A. F. Bakker, and J. M. J. van Leeuwen. Molecular dynamics of the surface tension of a drop. *Journal of Chemical Physics*, 96(1):565–576, 1992.

[11] M. J. P. Nijmeijer, A. F. Bakker, C. Bruin, and J. H. Sikkenk. A molecular dynamics simulation of the lennard-jones liquid-vapor interface. *Journal of Chemical Physics*, 89(6):3789–3792, 1988.

[12] E. M. Blokhuis and D. Bedeaux. Pressure tensor of a spherical interface. *Journal of Chemical Physics*, 97(5):3576–3586, 1992.

[13] H. E. Bardouni, M. Mareschal, R. Lovett, and M. Baus. Computer simulation study of the local pressure in a spherical liquid-vapor interface. *Journal of Chemical Physics*, 113(21):9804–9809, 2000.

[14] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, Oxford, 1987.

[15] D. Frenkel and B. Smith. *Understanding Molecular Simulation: From Algorithm to Applications*. Academy Press, San Diego, 1996.

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

[17] G. A. Chapela, G. Saville, S. M. Thompson, and J. S. Rowlinson. Computer simulation of a gas-liquid surface. *Journal of Chemical Society Faradays Trans. II*, 8:1133–1144, 1977.

[18] E. M. Blokhuis, D. Bedeaux, C. D. Holocomb, and J. A. Zolloweg. Tail correction to the surface tension of a lennard-jones liquid-vapor interface. *Molecular Physics*, 85(3):665–669, 1995.

[19] C. D. Holcomb, P. Clancy, and J. A. Zollweg. A critical study of the simulation of the liquid-vapor interface of a lennard-jones fluid. *Molecular Physics*, 78(2):437–459, 1993.

[20] M. Meche, J. Winkelmann, and J. Fischer. Molecular dynamics simulation of the liquid-vapor interface: The lennard-jones fluid. *Journal of Chemical Physics*, 107(21):9264–9270, 1997.

[21] M. Meche, J. Winkelmann, and J. Fischer. Molecular dynamics simulation of the liquid-vapor interface: Binary mixtures of lennard-jones fluids. *Journal of Chemical Physics*, 110(2):1188–1194, 1999.

[22] J. Trokhymchuk, A.; Alejandre. Computer simulations of liquid/vapor interface in lennard-jones fluids: Some questions and answers. *Journal of Chemical Physics*, 111(18):8510–8523, 1999.

[23] L. J. Chen. Area dependence of the surface tension of a lennard-jones fluid from molecular dynamics simulations. *Journal of Chemical Physics*, 98(23):10214–10216, 1995.

[24] J. G. Weng, S. Park, J. R. Lukes, and C. L. Tien. Molecular dynamics investigation of thickness effect on liquid films. *Journal of Chemical Physics*, 113(14):5917–5923, 2000.

[25] S. Kawano. Molecular dynamics of rupture phenomenon in a liquid thread. *Physical Review E*, 58(4):4468–4472, 1998.

[26] M. Moseler and U. Landman. Formation, stability and breakup of nanojets. *Science*, 289:1165–1169, 2000.

[27] E. D. Herrera, J. Alejandre, G. R. Santiago, and F. Forstmann. Interfacial tension behavior of binary and ternary mixtures of partially miscible lennard-jones fluids: A molecular dynamics simulations. *Journal of Chemical Physics*, 110(16):8084–8089, 1999.

[28] J. Stecki and S. Toxvaerd. The liquid-liquid interface of simple liquids. *Journal of Chemical Physics*, 103(10):4352–4359, 1995.

[29] M. Meyer, M. Mareschal, and M. Hayon. Computer modeling of a liquid-liquid interface. *Journal of Chemical Physics*, 89(2):1067–1073, 1988.

[30] Y. Shiraichi and Y. Hagiwara. Molecular dynamics simulation for evaporation at the interface between two immiscible liquids. *Proc.* 5^{th} *ASME/JSME Joint Thermal Engineering Conference*, pages AJTE–6515(1–6), 1999.

[31] F. Bresme and N. Quirke. Computer simulation studies of liquid lenses at a liquid-liquid interface'. *Journal of Chemical Physics*, 112(13):5985–5990, 2000.

[32] J. A. Barker. Surface tension and atomic interactions in simple liquids argon, krypton and xenon. *Molecular Physics*, 80(4):815–820, 1993.

[33] J. Alejandre, D. J. Tildesley, and G. A. Chapela. Fluid phase equilibria using molecular dynamics: The surface tension of chlorine and hexane. *Molecular Physics*, 85(3):651–663, 1995.

[34] S. Maruyama, T. Kurashige, and Y. Yamaguchi. Molecular dynamics of liquid structure near solid-liquid interface. *Proc.* 33^{rd} *National Heat Transfer Symposium of Japan*, 1:317–318, 1996. In Japanese language.

[35] S. Maruyama, T. Kurashige, S. Matsumoto, Y. Yamaguchi, and T. Kimura. Liquid droplet in contact with a solid surface. *Microscale Thermophysical Engineering*, 2:Liquid Droplet in Contact with a Solid Surface, 1998.

[36] S. Matsumoto. Molecular dynamics study of diffusion of liquid molecules near the three-phase contact. *Proc. AIAA/ASME joint Thermophysics and Heat Transfer Conference*, 3:179–184, 1998.

[37] T. Kimura and S. Maruyama. Molecular dynamics simulation of nucleation of liquid droplet on solid surface. *Thermal Science and Engineering*, 8(5):7–13, 2000. In Japanese Language.

[38] K. Nakabe, K. Yamanaka, and K. Suzuki. An md simulation on micro droplet behaviors affected by solid surface conditions. *Proc. of the* 5^{th} *ASME/JSME Joint Thermal Engineering Conference*, pages AJTE–6509(1–8), 1999.

[39] G. Saville. Computer simulation of the liquid-solid-vapor contact angle. *Journal of Chemical Society, Faradays Transactions II*, 73(2):1122–1132, 1977.

[40] G. Navascués and M. V. Barry. The statistical mechanics of wetting. *Molecular Physics*, 34(3):649–664, 1977.

[41] J. H. Sikkenk, J. O. Indekeu, J. M. J. van Leeuwen, E. O. Vossnack, and A. F. Bakker. Simulation of wetting and drying at solid-fluid interfaces on the delft molecular dynamics processor. *Journal of Statistical Physics*, 52(1/2):23–44, 1988.

[42] M. J. P. Nijmeijer, C. Bruin, A. F. Bakker, and J. M. J. van Leeuwen. Wetting and drying of an inert wall by a fluid in a molecular-dynamics simulation. *Physical Review A*, 42(10):6052–6059, 1990.

[43] J. Z. Tang and Harris J. G. Fluid wetting on molecularly rough surfaces. *Journal of chemical Physics*, 103(18):8201–8208, 1995.

[44] P. P. Ewald. Die berechung optisher und electrostatisher gitterpotentiale. *Annual Physics*, 64:253–287, 1921.

[45] S. W. de Leeuw, J. W. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. i. lattice sums and dielectric constants. *Proc. of Royal Society of London, A*, 373:27–56, 1980.

[46] S. W. de Leeuw, J. W. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. ii. equivalence of boundary conditions. *Proc. of Royal Society of London, A*, 373:57–66, 1980.

[47] M. Deserno and C. Holm. How to mesh up ewald sums. i. a theoretical and numerical comparison of various particle mesh routines. *Journal of Chemical Physics*, 109(18):7678–7701, 1998.

[48] T. Darden, D. York, and L. G. Pedersen. Particle mesh ewald: An $n\log(n)$ method for ewald sums in large systems. *Journal of Chemical Physics*, 98:10089–10092, 1993.

[49] H. J. C. Berendsen, J. P. M. Postma, W. F. Van Gunsteren, A. Di Nola, and Haak J. R. Molecular dynamics with coupling to an external bath. *Journal of Chemical Physics*, 84(8):3684–3690, 1984.

[50] J. N. Israelachvili. *Intermolecular and Surface Forces.* Academic Press, San Diego, 1992.

[51] H.C. Hamaker. *Physica*, 8:1058–1072, 1937.

[52] F. London. *Trans. Faraday Society*, 33:8–26, 1937.

[53] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles.* Adem Hilger imprint by IOP Publishing Ltd., Bristol, 1988.