STUDY OF PROPERTIES AND BEHAVIOR OF SURFACTANTS AND MICELLES AT THE SOLID/LIQUID INTERFACE USING MOLECULAR DYNAMICS SIMULATIONS

By

KUNAL SHAH

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2005

This document is dedicated to my parents with love and gratitude.

ACKNOWLEDGMENTS

First of all, I would like to acknowledge my advisor, Dr. Susan Sinnott, for her support and guidance throughout my PhD. research work. Her faith in me helped me through the tough time in my research and also was a constant source of inspiration. Her openness and scientific expertise have made working in her group a pleasant and learning experience for me. I would also like to thank Dr. Brij Moudgil for his supervision and suggestions towards my research work. His passionate attitude towards research made me a competitive person in life. I would like to thank Dr. Dinesh Shah for his advice and motivation in both personal and professional life. I would also like to thank Dr. Wolfgang Sigmund, Dr. Laurie Gower, and Dr. José Fortes for their support throughout my research work.

Special thanks go to Patrick Chiu for being a wonderful colleague and a friend. I have always enjoyed the times that we have worked together on making this project successful. I would also like to thank my past group members, Dr. Doug Irving, Dr. Yanhong Hu, Dr. Inkook Jang, Dr. Ki Ho Lee, and Dr. Boris Ni for their encouragement and support. I also want to extend my gratitude towards my present group members for their friendly support and helpful discussions. Thanks go to Mayank Jain and Dr. Kequing Fa for their fruitful discussions and help towards my research.

I would like to thank my sister and brother-in law for their affection and support; my nephew, Jash whose love made me smile through the toughest times. I would especially like to thank Shruti for her help, support, and inspiration in life. I would like to thank Jairaj for being a great friend. My heartfelt appreciation goes to my parents for their love and support without which I would have not come this far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

STUDY OF PROPERTIES AND BEHAVIOR OF SURFACTANTS AND MICELLES AT THE SOLID/LIQUID INTERFACE USING MOLECULAR DYNAMICS SIMULATIONS

By

Kunal Shah

May 2005

Chair: Dr. Susan B. Sinnott
Major Department: Materials Science and Engineering

Dilute and concentrated surfactant systems at the solid-liquid interface are examined using classical molecular dynamics simulations. Particular emphasis is placed on understanding how surfactants aggregate and form the micellar structure, how micelles change shape at high concentrations in aqueous media and in the presence of hydrophilic and hydrophobic surfaces, and at what force this micellar structure breaks apart during indentation of micelle-covered surfaces with a proximal probe microscope tip. The specific system of interest is C_{12}TAB (n-dodecyltrimethylammoniumbromide) surfactant in an aqueous medium that is modeled with empirical potentials.

The simulations predict that the micelle structure in water is compact and either spherical or elliptical in shape. In the presence of a hydrophilic surface of silica, the structure evolves into a flat elliptical shape, in agreement with experimental findings. In the presence of hydrophobic surface of graphite, the aggregate evolves in to hemi-cylindrical structure with tails of surfactants lying on the surface due to hydrophobic interaction. This finding is in agreement with experimental data.

The simulated indentation of the micelle/silica system causes the micelle to break apart at an indentation force about 1 nN and form a surfactant monolayer. The predicted force curve is in excellent agreement with experimental measurements. The simulated indentation of micelle/graphite system causes breakage of micelle at an indentation force of about 1.25 nN, which is slightly above the force predicted to break the micelle structure on silica (1 nN). This difference can be explained by a stronger interaction (hydrophobic) between the absorbed structure and the graphite substrate.

CHAPTER 1
INTRODUCTION AND BACKGROUND

1. 1 Introduction

Amphiphilic systems are of significant interest to researchers in both scientific and industrial fields. Surfactant systems are used as organic templates in the synthesis of inorganic materials with nanoscale porosity. These systems that occur at solid/liquid interfaces[1] are important in a wide range of industrial processes, from paint industries and ore floatation to enhanced oil recovery. [2]

Surfactant structures at the solid/liquid interface are increasingly being utilized as dispersants at high electrolyte concentrations, where conventional dispersants, e.g., inorganic dispersants and polymers, may not perform well. [2] For example, in the chemical mechanical polishing of silicon wafers, nanometer-scale silica particles are used as polishing media. The presence of even small aggregates of these particles can create scratches on the surface and damage the silicon wafer. Thus, it is important to maintain a uniform dispersion of silica particles. In the case of nanoparticles, maintenance of dispersion becomes challenging because of inadequate knowledge and understanding of dispersion and adsorption. Hence, the development of better performing dispersants requires a fundamental understanding of the adsorption mechanism of dispersants on the solid at the molecular level.

In many industrial processes, such as chemical mechanical polishing, mixing and flow of particulate suspensions, self-assembled structures of surfactants encounter appreciable loads and pressures. To design process conditions for optimal results, the

mechanical properties of these aggregates must be controlled. The force required to disintegrate the micelles is the upper limiting applied load for dispersion. Above this load, the aggregate micelle structure is disintegrated and undesirable adhesion occurs between the two sliding surfaces. Over the last decade, several groups have used Atomic Force Microscopy (AFM) to study the mechanical properties of micelles. [3,4] The advantage of using AFM is that the load required to break the micelles apart can be directly measured. The disadvantage of this approach is that the mechanisms involved in micelle disintegration cannot be resolved.

1.1.1 Introduction of Surfactant

Surfactants are amphipathic, which means they have both a hydrophobic tail and a hydrophilic head group. In aqueous media, surfactants aggregate together and form a self-assembled structure called a micelle. Depending on the nature of the hydrophilic group, they are classified as anionic (negatively charged), cationic (positively charged), zwitterionic (both positively and negatively charged), and nonionic (not charged). At moderate concentrations these surfactants form spherically shaped micelles, while at high concentrations or in the presence of an electrolyte, they form cylindrical micelles. At the solid/liquid interface, the self-assembly process is influenced by water-surface interactions and, most importantly, surfactant-surface interactions. Depending on the nature of the surface, the properties of the surfactant, and the concentration of electrolyte in the aqueous media, the structure of the surfactants or the self-aggregate at the interface is a function of the concentration of surfactants in the system. At low concentrations, the surfactants adsorb randomly. At moderate concentrations hemi-micelle structures are seen, while at high concentrations, bilayers, spherical micelles, or cylindrical micelles are found. It is known that self-assembly of surfactants occurs at lower concentrations at

solid surface interfaces than in the aqueous bulk [5] having lower critical micelle concentration (cmc) values. This is due to adsorption of surfactants at the interface through hydrophobic or hydrophilic/electrostatic interaction.

1.1.2 Thermodynamics of Surfactants Adsorption on Surfaces

The adsorption free energy of surfactants at the solid-liquid interface, ΔG_{ads}, is calculated as follows:[6]

$$\Delta G_{ads} = \Delta G_{elec} + \Delta G_{h\text{-}h} + \Delta G_{chem} + \Delta G_{C\text{-}C} + \Delta G_{C\text{-}S} + \Delta G_{solv/desolv} \quad (1\text{-}1)$$

where ΔG_{elec} represents the electrostatic contribution, $\Delta G_{h\text{-}h}$ represents the contribution from hydrogen bonding, ΔG_{chem} represents the contribution from any chemical interaction between the solid surface and the surfactants, $\Delta G_{C\text{-}C}$ represents the contribution from any lateral interactions between the adsorbed hydrocarbon chains, $\Delta G_{C\text{-}S}$ represents contributions from the interaction of chains with nonpolar surfaces, and $\Delta G_{solv/desolv}$ represents energy from the solvation/desolvation of the surface and surfactant species upon adsorption.

Depending on the nature of the surfactant, the properties of the surface, solution properties, and electrolyte concentration, these terms vary in their importance in controlling the adsorption of surfactants on a solid surface and ultimately affecting the equilibrium structure of self-aggregates on the surface. However, in general, the electrostatic (ΔG_{elec}) and lateral chain interaction ($\Delta G_{C\text{-}C}$) energies are the major factors in determining the structure and morphology of surfactant aggregates on solid surfaces. For example, in the case of a cationic surfactant adsorbed on a negatively charged hydrophilic surface, initially, the electrostatic interaction between the head group of the surfactant and the negatively charged sites on the solid surface dominate. Beyond a certain amount of adsorption, chain-chain interaction (hydrophobic interaction) plays a

dominant role in furthering the adsorption process. However, in the case of a surfactant on a hydrophobic surface, the surfactant chain and hydrophobic surface interactions play a major role.

1.2 Review of Experimental Work on Cationic Surfactants

1.2.1 Cationic Surfactants in the Bulk

Cationic surfactants form aggregates in the aqueous medium at a particular concentration of surfactants in the system, which is the cmc. A variety of techniques have been used to characterize the molecular weight or aggregation number (number of surfactants in the micelle) of micelles. The most commonly used techniques are elastic light scattering, quasi-elastic light scattering (QELS), membrane osmometry, small angle neutron scattering (SANS), and fluorescence probing.[7] The first two methods can measure micelle sizes only at cmc's owing to their sensitivity to intermicellar interactions, while the other three methods can provide information at all the given surfactant concentration. Fluorescence quenching technique appears to be the most reliable and easiest approach for obtaining the aggregation number of micelles under any given experimental conditions.[7] The aggregation number, N, of cetyltrimethylammoniumbromide (CTAB) [8] micelles have been measured as a function of alkyl chain length. N values measured at the cmc for CTAB are consistent with spherical or near-spherical micelle structures. Quasi-elastic light scattering data[9-11] confirm this prediction. An increase in temperature results in a decrease in micelle aggregation numbers.[12-14] The organization of alkyl chains in spherical cationic micelles has been studied by NMR and SANS. The order parameter decreases rapidly going from head groups to terminal CH_3 groups suggesting liquid-like behavior.[15] This suggests very random organization of tail groups within spherical micelle. The internal motion of alkyl

chains is very rapid.[16] SANS results have shown that the hydrophobic core of a micelle is devoid of water.[17] Increasing surfactant concentration results in extensive micellar growth. CTAB micelles undergo very rapid growth in micelle size from spherical to rodlike structure.[18]

There are two models proposed for growth of a micelle in the bulk. If the growth of the micelle occurs by the adsorption of multiple surfactants (dimers, trimers, oligomers) followed by stepwise adsorption of monomers, the process follows the Smoluchowski kinetic model for growth of a micelle.

$$C_r + C_s \rightleftharpoons C_{r+s} \quad (1\text{-}2)$$

where C_r and C_s are micellar cluster of aggregation number r and s respectively. This model is valid when the system is far from equilibrium or the concentration of surfactant is much higher than the cmc because it leads to a faster rate to equilibrium of the system. When the growth of the micelle occurs due to step-wise addition of monomer into the cluster then it follows the Becker-Döring kinetic model.

$$C_r + C_1 \rightleftharpoons C_{r+1} \quad (1\text{-}3)$$

where C_r is micellar cluster of aggregation number r and C_1 is a single monomer. This model is valid when the system is closer to equilibrium or surfactant concentration is lower than cmc.[19,20]

1.2.2 Cationic Surfactant Adsorption at Solid-Aqueous Interface

The adsorption of solute at the solid-liquid interface results in an increase in local concentration of solute at the surface. When the interaction is favorable, the local concentration of solute will increase relative to the bulk solution. This is commonly referred as surface excess. The fast kinetics associated with the formation of surfactant

aggregates at the interface has prevented accurate investigation of reaction kinetics and reaction mechanisms. In addition, achieving full understanding has been further hampered by speculations over the equilibrium structures of adsorbed surfactants. In many cases, it has been misinterpreted as due to simple monolayers or bilayers rather than discrete surfactant aggregates that are present in many surfactant substrate systems.

To summarize, experimental limitations associated with kinetics and structural information hinder satisfactory determination of the mechanisms behind surfactant adsorption at solid-liquid interfaces. The next sections discuss some of the techniques that have been used to study equilibrium and kinetic information in order to describe the adsorption process.

1.2.2.1 Adsorption isotherms

The solution depletion method provides adsorption isotherm data. The adsorption isotherms are interpreted by looking at changes in the rate of increase in the surface adsorbed surfactant with the concentration of surfactants. The equilibrium morphology/structure of self-assembled surfactant structures has been studied using adsorption isotherms.[4,21,22]

This adsorption isotherm data have been used to propose three theories (the Bilayer model, the two-step model, and the four-step model) for cationic surfactants adsorption occurs on negatively charged surfaces. Each theory proposes different structures or morphologies of self-assembled surfactant structures occurring at the surface with respect to the surfactant concentration in the system.

According to the bilayer model, local bilayer patches of surfactants form at a critical concentration without forming hemi-micelles on the surface. At high concentrations these patchy bilayers form a continuous bilayer.[23]

According to the two-step model (see Fig. 1-2), the surfactant adsorbs via electrostatic interaction with a silica substrate in region I. In region II the substrate charge has been neutralized. The surfactants are still adsorbed in the form of monomers. The monomers electro-statically adsorbed in region II are thought to be anchor points for spherical micelle formation. In region III there is an abrupt increase in adsorption in HMC (hemi-micelle concentration), which signifies the onset region in which surfactant concentration is sufficient to lead the hydrophobic interaction between monomers.[24] In the fully spherical micelle the surfactant concentration is not adequate and one can expect more adsorption of surfactant at those points. In region IV, the concentration is greater than the cmc, and is indicative of the formation of fully formed aggregates. [15, 22, 24-26]

DTA^+ adsorption on silica shows a typical two step model pattern (see Fig. 1-1)

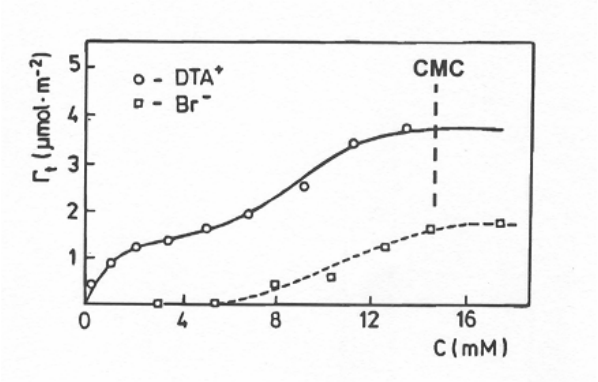


Figure 1-1: Adsorption isotherms of DTA+ (O) and Br- (□) ions measured on particulate silica at pH 8. Note the adsorption of bromide ions only occurs once the second increase has commenced for the DTA+ ions. The solid and the dashed lines were drawn by hand to guide the eye. The solution cmc is indicated by a dashed vertical line. [27]

The two-step model proposed by Gu [22] seems to be invalid for describing lateral hydrophobic interactions because it fails to account for the increase of surface charge that occurs in the second region of the isotherm.

Somasundaran [28] proposed a four-step model or the reverse orientation model. This method has been particularly successful in determining the adsorption behavior on alumina and rutile.

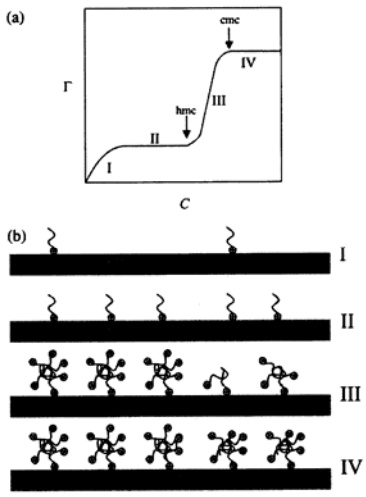


Figure 1-2: The two-step model for cationic surfactants adsorbed on silica (a) the general shape of the adsorption isotherm. The x axis indicates residual surfactant concentration anfd the y axis indicates adsorption density. (b) The proposed model of adsorption. [24]

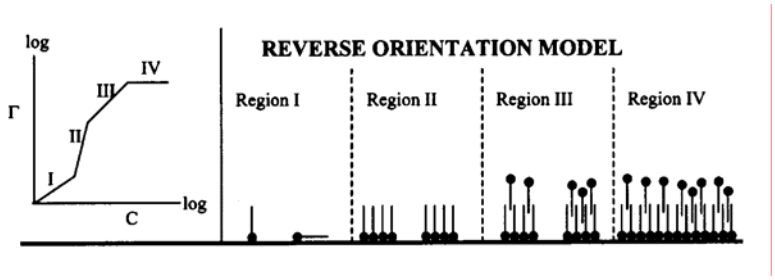


Figure 1-3: The four regions of the reverse orientation model of adsorption. Proposed adsorption isotherm and surfactant aggregates on solid substrates. Adapted from [21]

As seen from Fig. 1-3, in region I, the surfactant monomers are electro-statically adsorbed on the surface. Region II shows strong lateral interaction between adsorbed monomers and the formation of hemi-micelle. [21,29,30] Region III shows bilayer formation and also neutralization of surface charge. Region IV shows a fully formed bilayer.

Singh's [2] work with indirect methods, like Fourier transform infrared absorption (FT-IR), contact angle measurements, and surface force calculation, shows that the C_{12}TAB surfactants do not form bilayers in the presence of silica surface; surfactants form hemi-micelle to fully spherical micelle with increasing surfactant concentration in the system.

1.2.2.2 Atomic force microscopy

Perhaps the greatest advancement for the study of surfactant adsorption at the solid-liquid interface is the discovery and use of AFM. With this technique, *in situ* images of the adsorbed aggregates on the surface can be obtained. These data allow adsorption isotherms to be analyzed with knowledge of the morphology of these aggregates and are complementary to neutron reflectivity results to great accuracy. AFM is very well suited in studying the periodicity of discrete adsorbed aggregates on the surface and thus peak to peak separation can be measured. However, AFM is not without any limitations. This technique is only suitable when head-groups of surfactants are aligned towards solution to provide repulsive forces. Thus meaningful data can only be obtained when the concentration of surfactant is above the cmc. In the case of bilayers, the nature of the underlying layer remains speculative. AFM does not provide any information about surface excess or density of adsorption between aggregates. Thus, several morphologies can be consistent with AFM images, particularly in the case of bilayer aggregates. However, when AFM results are analyzed with other forms of adsorption data, the most likely structure can be justified with reasonable confidence. Thus, AFM is most useful when used in conjunction with other measurements.

The technique involves controlling the interaction between a tip attached to a cantilever and a flat substrate mounted on a piezoelectric crystal. The piezoelectric

crystal enables the movement of substrate in the vertical and lateral directions. The cantilever is made up of silicon and has a bypyramidal tip at the end. The radius of the tip is usually on the order of 10 nm[31].

The measurement of force on hard surfaces is done using "contact mode." In this mode, the substrate is brought to the tip, which is attached to the cantilever. Depending on the interaction between the surface and the tip, the cantilever either bends away (repulsive forces), or bends towards the surface (attractive forces). The deflection of the cantilever is detected through position sensitive photodiode detectors. Measurements are repeated in all three directions (x, y and z) to get an entire image. This mode is used to image substrates such as mica, oxides, and graphite in which bringing the substrate to the tip does not alter the nature of the substrate.

The most commonly used mode for imaging adsorbed surfactant aggregates on solid substrates is "soft non-contact mode."[32] In this technique, the interaction between surfactant aggregate adsorbed on the substrate and AFM tip is calculated.

AFM experiments are extremely useful in understanding different surfactant aggregate morphologies accomplished by different solution conditions or surface modifications. The size, shape, and spacing of the adsorbed structure are dependant upon intermolecular and surfactant-substrate interactions,[1] solution condition and type of surfactant.

CTAB surfactant adsorption on graphite

The first direct imaging of cationic CTAB surfactants adsorption at the interface was done on graphite surface.[33] Graphite is a frequently used substrate for AFM because it is available in an atomically smooth crystalline form that is ideal for AFM

investigations. In aqueous media the graphite substrate acts as a hydrophobic adsorbate. So, the contact between the surfactant and graphite is primarily through hydrophobic interactions between the surfactant tail and the graphite substrate. As shown in Fig. 1-4a parallel stripes spaced 4.2 nm are seen at CTAB concentrations between 0.8 and 5 mM. The orientation is perpendicular to the symmetry axis of the substrate as shown in Fig. 1-4b. It was concluded from the AFM images that the confirmation of aggregate was a hemi-cylindrical arrangement, as shown in Fig 4b. This can be correlated as a two-step adsorption isotherm.[34] For concentrations lower than the cmc, alkyl chains of surfactant extend in the plane of substrate and, as the concentration increases, monomers interact laterally and orient themselves perpendicular to the substrate. This was predicted by Manne et al. using AFM technique.[33]

Surfactant monomers bind very strongly to the graphite surface via hydrophobic interactions forming a film that has very low exchange rate with solution surfactants. Graphite exhibits the highest degree of control over adsorbed structures compared to any other substrate due to the very strong hydrophobic interaction between the surfactant tail and water and the graphite surface and water. Hemi-cylindrical aggregation has been observed on graphite for ionic,[1,33,35-37] non-ionic, [32,38,39] and zwitter-ionic[40] surfactants having more than 12 carbon atoms in their tail.

CTAB surfactant adsorption on mica

Different aggregate morphologies have been determined in the presence of mica surface. As mica is hydrophilic, the head group of surfactant interacts strongly to the surface. The area of interaction per surfactant molecule is therefore significantly less than in the case of graphite. The density of the charges on the mica surface is such that

surfactants are adsorbed to the surface with their head group closer to each other than is the case in the aqueous bulk.

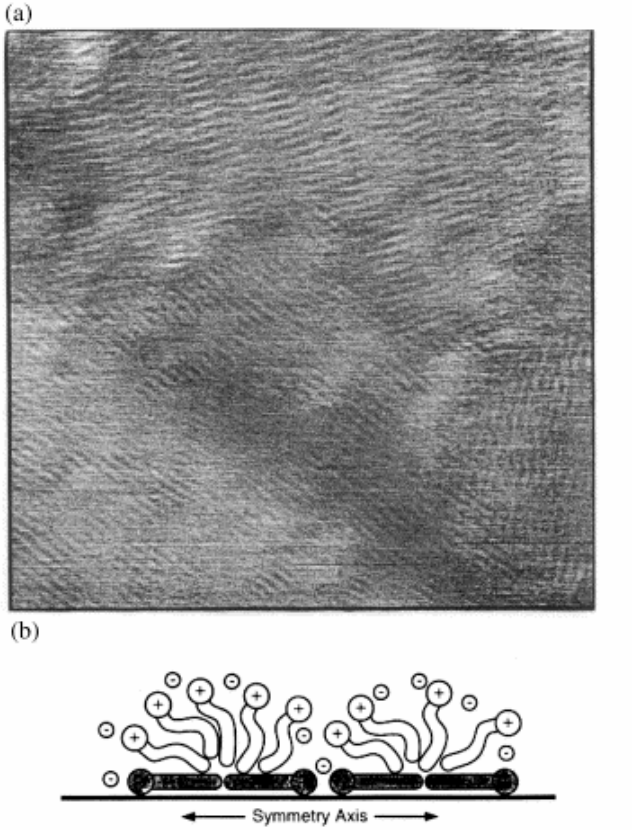


Figure 1-4: Adsorption of CTAB on graphite (a) an AFM image (b) proposed hemicylindrical structure of CTAB on graphite[33]

This leads to a lower degree of curvature for the aggregate than in the case of solution micelles. The negative charge sites on mica are arranged precisely on the surface lattice, which influences aggregate morphologies. The CTAB surfactants have been shown to form flattened cylindrical aggregates on mica. The period of these aggregates is similar to the diameters of micelles in solution but the period length is much larger.[1,41,42] The long axis of neighboring aggregates is aligned locally due to the crystallinity of substrate, which gives the surface a striped appearance as shown in Fig. 1-5.

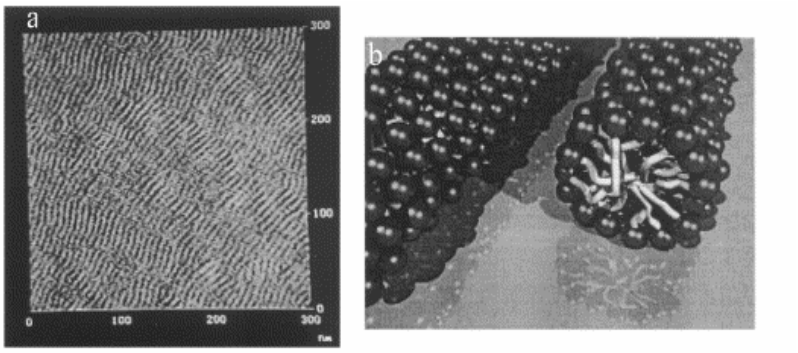


Figure 1-5: Adsorption of CTAB on mica (a) AFM image (b) Schematic representation of cylindrical structure of adsorbed CTAB on mica[42]

CTAB surfactant adsorption on silica

Relatively few AFM studies have been performed for surfactant adsorption on amorphous silica due to difficulty in obtaining clear images. In general, cationic surfactants have been shown to form spherical admicelles with no long range ordering on silica substrates.[1,43-45] This morphology agrees well with the mechanism predicted by traditional adsorption isotherms. For CTAB on silica, when surfactant concentration is increased from 0.9 cmc to 10 cmc, the adsorbed micelle morphology changes from short rods to worm-like structures. This study was done by Velegole et al. [43] and the results are shown in Fig. 1-6a and 1-6b.

Summary of AFM experiments

AFM has provided evidence for the presence of discrete aggregates at solid-liquid interfaces and this aids in the interpretation of surfactant adsorption at interfaces. AFM imaging is only useful above the concentration at which surfactants form aggregates at the surface. In these aggregates, surfactant head groups are facing towards solution. This makes it possible to impart a repulsion that allows soft contact imaging to be done. The hydrophobic graphite orients adsorbed structures more strongly than any other surface

due to very strong interaction with the adsorbing monomer, which leads to hemi-cylindrical structures.

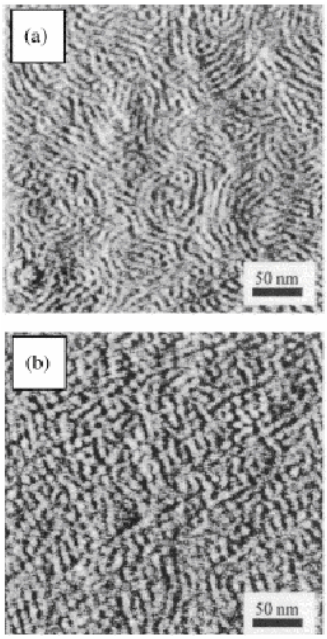


Figure 1-6: AFM images of CTAB surfactants adsorbed on silica (a) CTAB 10 × cmc (b) CTAB 0.9 × cmc[43]

The hydrophilic crystalline mica orients adsorbed structures periodically but not as strongly as graphite. CTAB surfactants form cylindrical aggregates. Surfactants are less strongly oriented on silica surfaces due to the lower surface charge and there is no long range ordering of the aggregates due to the amorphous state of the substrates. CTAB surfactants have been shown to spherical, flattened, admicelle structures with short rods at 0.9 cmc, and worm-like structures at 10 cmc.

Fluorescence quenching experiments

This method is well accepted for determining aggregation number of micelles in the bulk solution.[7] The fluorescence quenching technique statistically analyzes the fluorescence emission of micelle bound probe molecules. A second molecule, called a quencher, suppresses this emission. The probe, typically pyrene, is excited at a particular

wavelength and the fluorescence intensity is monitored at a second wavelength. All fluorescence quenching studies assume that there is no exchange between the aggregates and that the micelles in the system are unperturbed by the presence of the probe and quencher molecules. A random distribution of the probe and quencher throughout the system is assumed. So, some aggregates will have both a probe molecule and a quencher molecule, while some will have either a probe molecule or a quencher molecule, while other aggregates will have neither a quencher nor a probe molecule. Statistical analysis is used to calculate the average number of quencher molecules per micelle in the system. As only micelles containing probes are investigated this way, the aggregation number of the micelle can be determined by the intensity of fluorescence emission in the absence and presence of a known concentration of quencher. This approach is called the static emission method and for it to be successful quenching has to be rapid. It is assumed that there is no emission from micelles containing quenchers and thus, if there is any emission from these micelles, it will alter the aggregation number determined. This effect is expected to be pronounced for large micelles where the rate of quenching will be dependent on the diffusion rate of the quencher and probe molecules in the micelle core.

The time resolved fluorescence quenching experiment is not as dependent on the assumption of rapid quenching that we have made earlier. Rather, this experiment uses a delta pulse excitation of the probe, and decay of the emission with time is recorded. As the quenching rate is dependent on collisions between the quencher and probe, this gives fast initial decay followed by a slower unquenched decay. The procedure to determine the aggregation number is similar to the procedure for the static emission method.

To determine aggregation number, the concentration of surfactant must be known. At solution interfaces, the surface concentration must be determined by other methods. Also, it is assumed that all surfactants are aggregated in micelles. If aggregates are present both in the solution and at the interface, then the results are inaccurate in terms of determining the aggregation numbers. But, if the concentration of surfactants in chosen such that it is below the cmc of solution and above the concentration at which the surfactants form aggregates at interface, then only the adsorbed aggregate will be probed by the method and the aggregation number can be calculated precisely.

The first system to which the static emission method was applied was a non-ionic surfactant system. It was shown that the aggregation number for adsorbed surfactants had the same trend as bulk micelles.[46,47] Fan et al.[21] performed similar studies for CTAB surfactants on alumina substrates. The presence of small, surface adsorbed aggregates was determined (see Fig. 1-7). The size of these aggregates grows as the concentration of surfactant increases. This result agrees with the results of adsorption models for ionic surfactants in the presence of oppositely charged substrates suggested by adsorption isotherm data. In the model, surfactants initially adsorb electrostatically to the surface. They act as a nucleation site for further surfactant adsorption to the surface leading to the formation of an aggregate. These data also give insight into details of a structure of adsorbed surfactants aggregate at a concentration below the concentration at which a full aggregate is formed. This approach thus provides information that cannot be obtained using AFM. Strom et al.[48] used a time resolved fluorescence quenching method to study CTAB surfactants on silica surface. His data provided strong evidence for the formation of discrete aggregates at the silica surface; this finding agreed with AFM imaging results.

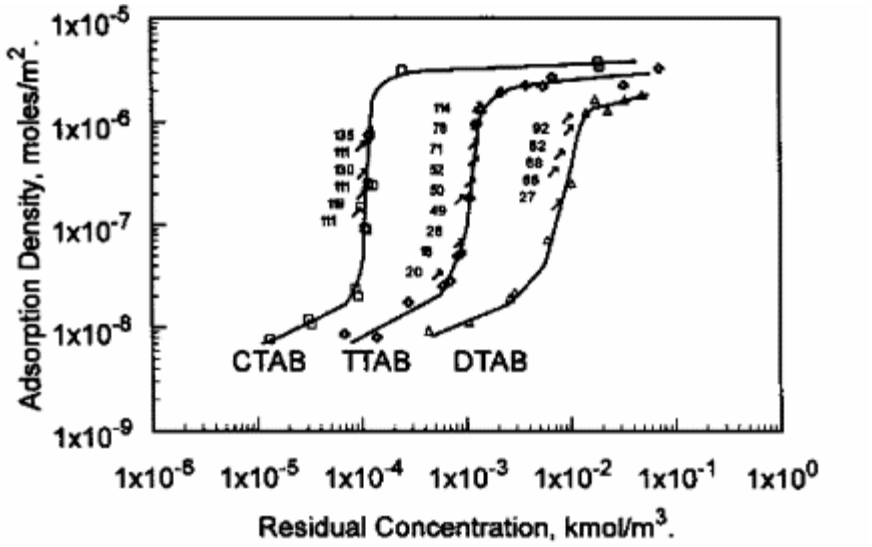


Figure 1-7: Aggregation number for adsorbed cationic surfactant aggregate on alumina of three different chain length.[21]

This method provides two important insights into surfactant-surface interactions. First, it provides strong evidence for the formation of discrete aggregates in good agreement with the findings of AFM. Secondly, several fluorescence quenching studies have shown increased aggregation numbers for adsorbed aggregates with increases in surfactant concentration in the system.

Reflectance techniques

The principle of reflectivity is similar to scattering regardless of light, x-rays, or neutrons being used as the radiation source. X-rays are sensitive to electron density and so investigate the atomic number of the species in the system. Neutron scattering is dependent on the nuclear properties of the system. The sensitivity of optical techniques like ellipsometry and optical reflectometry depend on reflection and refraction, which gives phase information. These effects allow calculation of adsorbed layer thickness or concentration of surfactant at the interface. Shorter wavelengths from x-rays and neutrons provide more thorough information about adsorbed layers.

Neutron Reflectivity (NR) provides sufficient resolution to study the orientation of surfactants adsorbed at the surface as different molecules scatter neutrons differently depending on their nuclei. This method is non- destructive and does not perturb surfactant morphology. By careful adjustment of proton to deuterium ratio of the solvent, the reflectivity profile can be made dependent only on the properties of the adsorbed layer. The resolution or the smallest length scale that can be probed is a function of a wavelength, which gives much higher resolution over optical techniques. With the use of contrast control, NR is very useful for studying concentration profiles close to the surface and orientation of adsorbed species at the surface.

The interpretation of NR results is model dependent. The adsorbed layer is not homogenous and much more complicated. Each layer is characterized by its thickness, scattering length density, and roughness factor. Scattering length density is dependent on surfactant head groups and alkyl chains.

Additional contrast provides information that aids the selection of appropriate models. Care needs to be taken in interpreting the data achieved by NR. If the limitations are taken into consideration useful information can be derived. It gives the most direct measurement of the thickness of adsorbed layers and provides information on the orientation of adsorbates. However, it does not provide any data related to the lateral ordering of the molecule. Hence one cannot use it to reach any final conclusions on the aggregate structure and dimensions.

Several studies have focused on adsorption of CTAB on silica.[49-51] They conclude that the surface is incompletely covered and it has been hypothesized that the CTAB forms either discrete micelle-like structures or defective bilayer structures. AFM and

fluorescence data have shown admicelle morphology on the surface. McDermott et al.[52] have shown that surface coverage increases with increases in concentration of surfactant. The layer thickness was 3.4 nm and it was further concluded that the aggregate structure is like islands of surfactants on the surfaces. It was also noted that head groups are protruding towards aqueous media in order to shield alkyl chains from water. Fragneto et al.[50] concluded that if the adsorbed aggregate was micelle-like then it must be strongly flattened. This result has been confirmed by several techniques and has now become well accepted. The surface preparation and surface roughness could also contribute to variation in aggregate structure on silica substrates. The adsorption of surfactant decreases as the nano-roughness increases. It has been noted that studies of adsorbed aggregate of surfactants at the interface have produced unequivocal results when two or more experimental techniques have been used. With the use of AFM imaging, Schultz et al.[53,54] have concluded discrete aggregates of surfactants form at interfaces with spherical and cylindrical structures. AFM and NR data are consistent with this conclusion provided that the model used to get NR data is appropriate.

Ellipsometry and Optical Reflectometry (OR)

Ionic surfactants initially adsorb at the surface electrostatically at low concentrations. As the concentration of surfactant increases, the lateral hydrophobic interaction leads to a hemi-micelle formation. At higher concentrations formation of aggregates can be seen. This interpretation is quite well established but whether these processes occur kinetically is still open to question. Even the formation of bilayers between hemi-micelle and fully formed aggregates is still questionable.

Surfactant adsorption kinetics is far less studied in contrast to equilibrium adsorption characteristics. It is very important to study adsorption kinetics in applications such as wetting, lubrication, spreading, etc. The time scales required to study adsorption kinetics is on the order of nanoseconds or millisecond to seconds. However recent development of ellipsometric techniques has made it possible to study the kinetics of adsorption and desorption.

OR and ellipsometry are techniques that monitor the variation in reflectivity upon adsorption with increases in concentration of surfactant. This variation is induced by changes in the reflective index of the substrate upon adsorption of surfactants. Both techniques use linearly polarized light that is reflected from the substrate. The polarization characteristic of reflected light is monitored. The changes in polarization are highly sensitive to the presence of surfactant layers. The extensive explanation of ellipsometry is given in reference [55] and for OR in reference.[56] It does not provide any information concerning the molecular structure of the adsorbed layer.

The cationic head group adsorbed at negatively charged sites on the silica surface and surface charge neutralization occurs at very low concentration of surfactants. Eskilsson and Yaminsky [57] used *in situ* ellipsometry to study CTAB surfactants at silica interfaces (see Fig. 1-8). As shown in the figure, the surfactant adsorption begins to increase significantly at approximately 0.1 mM and the most rapid increase is observed between 0.4-0.8 mM. At the beginning of the region of sharp increase, initial surfactant head groups adsorb on the surfaces and, once the surface is neutralized, more surfactants adsorb with head groups facing away from the surface. For concentrations above 0.4 mM the adsorbed layer thickness was approximately 2.5 nm. The kinetics of adsorption was

found to be dependent on the bulk concentration. When surfactant concentration was much less than the cmc it took up to 2 hours to reach to equilibrium at the surface. At the cmc adsorption was complete within minutes. For all surfactant concentrations the rate of desertion was relatively rapid. Pagec et al.[58] studied CTAB at silica using OR. Their results display similar kinetics to those obtained by Eskilsson and Yaminsky.[57]

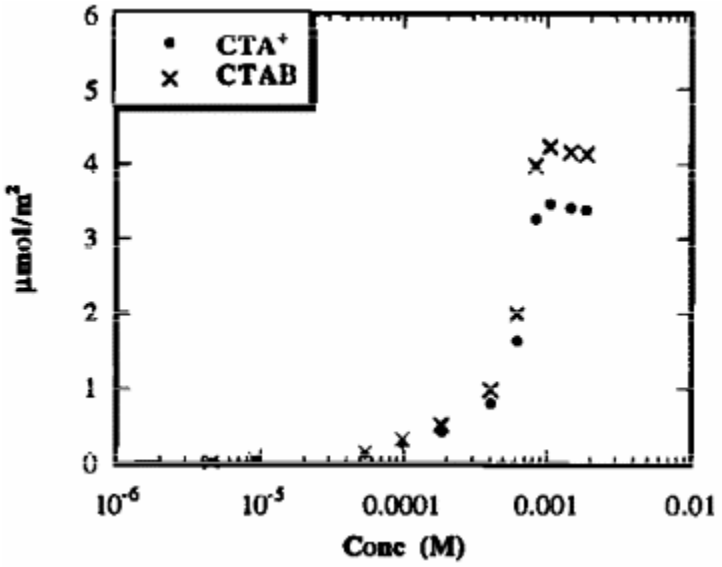


Figure 1-8: Adsorption isotherm of CTAB at silica-aqueous interface[57]

The sticking ratio of surfactants obtained with OR is shown in Fig. 1-9. The sticking ratio is seen to increase above the cmc. If surfactant molecules were competing for adsorption at the surface then we would assume a reduction in sticking ratio. But this study indicates that adsorption of surfactant monomers is cooperative. Cooperative adsorption is clearly aided by screening of head group charge by electrolytes. This screening plays a major role in allowing hydrophobic interactions to take over. This sticking ratio increases sharply at the cmc both in the presence and absence of electrolytes.

NR results have provided the most accurate measure of adsorbed layer thicknesses for CTAB on silica, providing values of approximately 3.5 nm. The lateral size of

discrete aggregates is 9 nm; these aggregates are flattened and are strongly adsorbed on the surface. These results agree well with AFM imaging data. Ellisometry and OR provide useful information about adsorption kinetics.

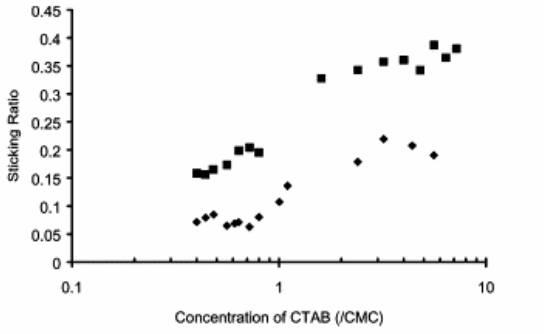


Figure 1-9: Sticking ratio vs. concentration for solutions of CTAB, normalized by the corresponding cmc[59]

1.2.2.5 Mechanism of adsorption

After achieving all the information available from the experimental techniques outlined above, the following figure (Fig. 1-10) is the proposed mechanism of surfactant adsorption at the liquid-solid interfaces for cationic surfactants on oxide surfaces. The adsorption is controlled by electrostatic/hydrophilic and hydrophobic interactions and the relative influence of this interaction is determined by the nature and concentration of surfactants and properties of surfaces like hydrophilic, hydrophobic, type of charge and density of charge.

The electrostatic concentration span

In the first span surfactant molecule adsorb at the interface electrostatically. The positively charged CTAB adsorb at negatively charged silica sites.

The electrostatic-hydrophobic concentration span

The surfactant tail groups interact with the hydrophobic regions present on the surface. Also, newly induced sites of adsorbed surfactant act as nucleation points for

further surfactant adsorption from the bulk to the surface. Thus the adsorption is driven by both the hydrophobic interaction and electrostatic attraction. Throughout the second span charge of the underlying substrate continues to increase and at the end the adsorbed morphology is hemi-micelle structure and overall surface charge is neutralized.

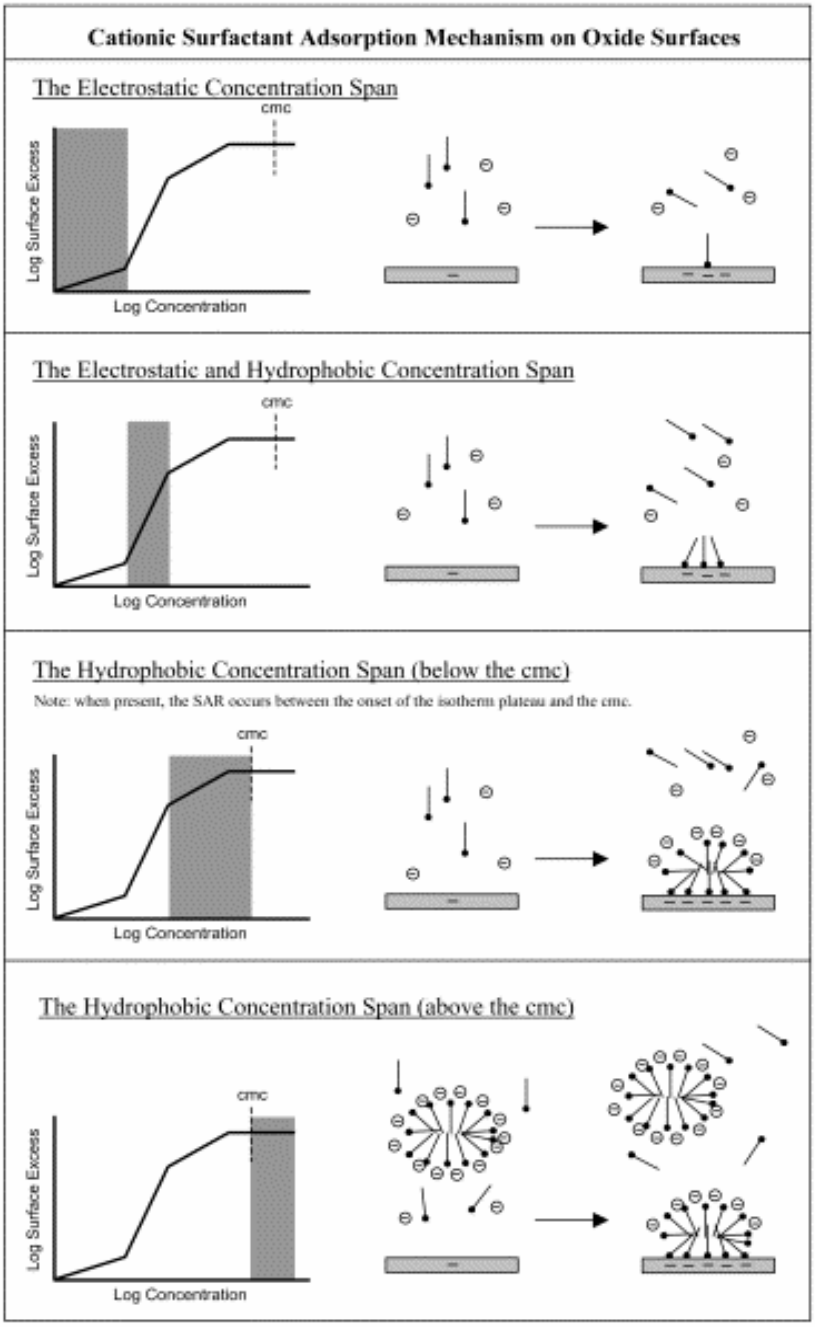


Figure 1-10: Proposed mechanism of cationic surfactant adsorption

The hydrophobic concentration span

Once the charge is neutralized in the second span, any further adsorption makes the interface positively charge due to adsorbing positively charged head group of CTAB. Any further adsorption will be against repulsive electrostatic barrier created by over compensation of surface charge by adsorbed surfactants and is now driven by purely hydrophobic interactions Thus this span represents purely hydrophobic adsorption of surfactant molecules with head groups of surfactants facing away from the surface. Above cmc adsorption of entire micelle on the substrate can also be seen.

1.2.2.6 Summary of experimental techniques

Adsorption isotherms are the traditional methods for investigating surfactant adsorption at interfaces. Data obtained with AFM, probe studies, NR, and ellipsometry provide valuable information concerning the confirmation of surfactants and the dimensions of the structure.

Valuable information about self-assembled surfactant structures has been provided by experimental techniques such as the solution depletion method for constructing adsorption isotherms,[4,21,22] electron spin resonance (ESR),[21] Raman spectroscopy,[60] fluorescence decay,[61] NR,[62] FT-IR,[2] surface force measurements,[2] and ellipsometry.[57] These techniques provide some information about hemi-micellization, cmc, and adsorption kinetics. AFM has also been used to study CTAB surfactants at concentrations above cmc.[1,43,45] It has been predicted that the CTAB surfactants with cationic head groups form spherical micelles on silica surfaces, cylindrical micelles on mica surfaces, and half cylindrical micelles on hydrophobic graphite surfaces. However knowledge of the morphology and structure of self-assembled micelle structures and the dynamics of

structural transitions from one shape to another still remains inadequate, as very few experimental techniques can directly probe these structures on the interface.

Techniques such as ellipsometry[57] and neutron reflectivity[62] can study the kinetics of adsorbed surfactant structures at the solid-liquid interface *in situ*. The minimum time required to collect data is in the order of 10-30 seconds whereas single monomer adsorption on the surface occurs on the order of picoseconds and micellar aggregation occurs on the order of milliseconds. Hence there is much about the micelle formation process that cannot be monitored experimentally. Thus, while experimental techniques on a whole can provide quantitative data about surfactant adsorption, they provide little information about the kinetics of processes that determine the topology, structure, shape, and mobility of the self-assembled structure formed at the solid surface.

1.3 Computer Simulations

All these experimental techniques provide quantitative data about the measure of surfactant adsorption. However, they provide little information about the topology, structure, shape, and mechanical properties of the self-assembled structure formed at the solid surface interface. Knowledge of the morphology of the adsorbed structure on the surface is still inadequate and better understanding is required at both molecular and atomic level. Computer simulations are in a niche position to provide the details that are lacking in the experimental data. Although this type of work requires long time-scales, the use of molecular dynamics (MD) simulations and the increasing power of computers in recent years make it possible to study the dynamics of micellar structure in the bulk and adsorption of surfactants at solid-liquid interface using an atomistic or pseudo-atomistic approach. Also, the parallelization of the MD program helps treat big systems (~100,000 atoms) and makes simulations run faster.

MD simulations numerically integrate Newton's equation of motion such that all the atoms or pseudo atoms in the system are allowed to evolve in time in response to the forces that act on them. They can therefore help explore phenomena such as equilibrium structure/morphology of self-assembled structures in the bulk and at the interface as well as the dynamics of the self-assembly process at the interface, tracking the transition occurring from one structure to another. As the time range in MD simulations is in the scale of femto seconds it will enable the visualization of self-assemble process at a molecule by molecule level at both low and high concentrations; experimental tools fail to do so.

Consequently, MD simulations are useful tools to explore aggregation number, preferred shape, and structure of the micelle as a function of surfactant concentration, electrolyte concentration, and chain length of the surfactants. They can also provide information about the mechanical properties and behavior of the micelles at solid surface interfaces. These simulations are therefore carried out in bulk and at solid surface interfaces with surfactant concentrations above cmc where aggregation of the surfactant is expected. Every simulation runs for a few hundred picoseconds to few tens of nanoseconds. Although these times scales are too short to entirely understand the true dynamic properties of micelles and surfactants (changes and fluctuations in the shape of micelles and micelle aggregation number can vary from 10^{-11} to 10^{4} seconds), the simulations can provide information which is representative of the processes occurring at the dynamic level. MD simulation can be used to perform simulations only in the range of several nano seconds. It cannot be used to trace activities occurring in the range of micro and milli seconds. However this does not limit the scope of this study, as in this

study we are starting the simulation with hemi-micelle, bilayer or spherical micelle already built in the system. Also, we are looking at the stability and evolution of self-assembled structure.

Because of the long time scale involved, very few computer simulation studies have focused atomistic simulations of aggregation of surfactant in bulk and at interface. In recent years, due to increasing power of computers and development of systematic methodologies and atomistic model,[63] it has become possible to study self-assembly of surfactants.[20,64,65] MD simulations have been used previously to study surfactant systems in the bulk[20] and at graphite surfaces.[65] However, to our knowledge, no studies have been done to study the morphology, structure, and mechanical properties of self-aggregating surfactants at the solid-liquid interface with hydrophilic surfaces (silica). In the present study, we use MD simulations to consider surfactant structures at the water-silica and water-graphite interface and simulate their indentation with silica and graphite AFM tips. The results are compared to experimental measurements.

1.4 Organization of Thesis

The second chapter presents an introduction of the MD method and a discussion of all the models and parameters individually. The third chapter discusses the simulations done only in aqueous media and discussing growth of micelle and favorable structure of micelle in the bulk. The fourth chapter presents results for the adsorption of micelles at solid-liquid interfaces and the difference in adsorbed micelle structure at hydrophobic and hydrophilic surfaces. Depending on the nature of the surface, different morphologies of micelles has been seen. Flat elliptical micelle can be seen at negatively charged hydrophilic silica surface and hemi-cylindrical micelle structure can be seen at nonionic hydrophobic graphite surface. The fifth chapter discusses the mechanical properties and

breakage characteristics of micelles at silica and graphite surfaces using silica and graphite indenters, respectively. The sixth chapter summarizes the results of the study and provides recommendations for future work.

CHAPTER 2
COMPUTATIONAL DETAILS

Computer simulations are in a niche position to provide the details that are lacking in the experimental data. The increasing power of computers in recent years make it possible to study the adsorption of surfactants at liquid-solid surface interfaces using molecular dynamics simulations with an atomistic or pseudo-atomistic approach. Also, the parallelization of MD programs allows the simulations to treat large systems that contain, on average, several 100,000-1,000,000 atoms and allows the simulations to proceed at a rapid enough rate to model phenomena that occur on the order of several nanoseconds.

2.1 Introduction to Classical Molecular Dynamics (MD) Simulations

The MD method was first introduced by Alder and Wainwright in the late 1950's [66,67] to study hard sphere systems. Since then MD methods have developed many new algorithms and tools. Now, MD can be used to study various system states, such as gases, liquids, surfaces, bulk defects, fracture phenomenon, friction, and biomaterial systems.[68] MD simulations numerically integrate Newton's equation of motion such that all the atoms or pseudo atoms in the system are allowed to evolve in time in response to the forces that act on them. MD simulation methods can be broken down into classical and quantum mechanical approaches. Quantum MD uses first principles approaches to calculate potential energies and forces acting on particles in the system under consideration. In contrast, classical MD uses empirically derived functions to calculate potential energies and forces acting on the particles in the system. Empirically derived

expressions for calculating potential energies and forces rely on fitting parameters that are set by comparison to experimental data or quantum mechanical calculations.

As discussed above, MD simulations numerically integrate Newton's equation of motion. Newton's second law states that the force vector, F_i, on particle i is the product of mass m_i and acceleration a_i as shown in Eq. 2-1.

$$F_i = m_i a_i \tag{2-1}$$

Before explaining how the new positions of each particle are calculated, some basic tools of MD are described below.

2.1.1Basic Concepts in MD Simulations

2.1.1.1 Hamiltonian

The Hamiltonian function, H, is the total energy of the system expressed in terms of the momenta and coordinates of the particles.[69] When a particle of mass m moves in one direction x then the classical Hamiltonian is defined as follows:

$$H= \frac{p_x^2}{2m} + V(x) \tag{2.2}$$

where p_x is the linear momentum of the particle in the x direction and V(x) is the corresponding potential energy of the particle.

2.1.1.2 Ensemble

An ensemble is an assembly of similarly prepared large number of systems. This collection of systems has the same macroscopic or thermodynamic properties but different microscopic properties. Ensembles are differentiated by keeping three thermodynamic quantities constant as shown in the table 2-1.

All the simulations in this study are done with canonical[70,71] ensembles that keep the temperature of the system constant with the velocity rescaling method. The

temperature typically fluctuates throughout the simulations around the set value by about 300 K.

Table 2-1: Types of ensembles and constant quantities. N is total number of particles in the system, V is the volume of the system, E is the total energy, P is the pressure, T is the temperature, and μ is chemical potential.

Ensemble	**Constant quantities**
Microcanonical	N, V, E
Canonical	N, V, T
Grand canonical	μ, V, T
Isobaric-isothermal	N, P, T

2.1.1.3 Kinetic energy and temperature

In terms of thermodynamic properties, Eq. 2-3 describes the state of a system, temperature T is defined with entropy S and energy E as follows:

$$1/T= (\partial S/\partial E)_{V,N} \quad (2\text{-}3)$$

Eq. 2-3 is appropriate for equilibrium states, but for non-equilibrium MD simulation, the definition of temperature is different. The average kinetic energy of the system is defined as follows:

$$< \tfrac{1}{2} mv^2> = k_B T/2 \quad (2\text{-}4)$$

where k_B is the Boltzman constant, m is the mass, and v is velocity. So Eq. 2-4 can be described as follows to calculate the temperature of non-equilibrium MD systems:

$$T= 1/k_B \sum(i=1 \text{ to } N) m_i v_i^2/N_f \quad (2\text{-}5)$$

where N is the total number of particles, i denotes the particle under consideration, and N_f is the number of degrees of freedom. N_f is 3N-3 for an N particle system.

2.1.1.4 Velocity rescaling

Velocity rescaling is a method for effectively controlling the temperature of the system to the desired temperature. We have used Berendsen method.[72] Before introduction of Berendsen method, it is worthwhile to discuss Anderson method first. Anderson method of temperature control proposed in 1980[73] can be thought of having the actual system coupled with thermal bath at the desired temperature. The coupling is simulated by the random collision between particles in the system to the particles in the thermal bath. After each collision, the velocity of the randomly chosen particles in the system is reset to the velocity corresponding to the desired temperature. The frequency of the random collisions is selected such that the thermal energy transfer between the system and the thermal bath is comparable to that in a system of same size embedded in an infinite thermal bath. This method is consistent with canonical ensemble and easy to use but it imparts drastic change to the system dynamics. Therefore, it is not appropriate to use Anderson method to study dynamical properties, although it is effective in studying static properties like density, pressure etc.

The more practical method is Berendsen which is appropriate method to study dynamics of the system.[72] Similarly as Anderson method, the system is coupled with external thermal bath held at the desired temperature T. However, the exchange of the thermal energy between system and the thermal bath is much smoother. Instead of the drastic resetting of the velocity of the system particles, the velocity of the particles are gradually scaled by multiplying it by a factor λ given by

$$\lambda = [1 + \Delta t/\tau_T(T/T_{ins} - 1)]^{1/2} \qquad (2\text{-}6)$$

where Δt is the time step, τ_T is the coupling time constant. In this method, the velocity of the particles is adjusted such that the instantaneous temperature of the system T_{ins}

approaches to the desired temperature T. Exchange of the thermal energy and the fluctuation in the energy of the system is controlled by the coupling time constant τ_T. If quick temperature is desired then coupling time constant is kept small thus λ becomes larger and change of velocity of particles in the system will be drastic. Dynamic properties of the system are greatly affected by the value of the coupling time constant. Berendsen et al. concluded with their testing that reliable dynamic properties can be achieved by a coupling time constant value greater than 0.1 ps. The advantage of the Berendsen method over Anderson method is the control of the value of the coupling time constant and thus the control over change of the system dynamics. The ratio $\Delta t/\tau_T$ in Eq. 2-6 is set to be 0.1 in our simulations because it gives best compromise between ideal temperature control and disturbance of the physical properties of the system.

2.1.1.5 Periodic boundary conditions and cutoff distances

MD simulations are useful in revealing atomic or molecular-scale details responsible for experimentally observed macroscopic system. However, computers still cannot model more than a few tens of millions of atoms, despite recent rapid advancements in computer power. These numbers are still much less than the number of atoms present in actual systems, which are typically in the range of 10^{23} atoms. Thus, in computational modeling the behavior at the system boundaries is an important issue. The states of the particles at boundaries are different from bulk states as they are deficient in bonding. In order to model a truly macroscopic system with a fixed number of particles, N, in the system, periodic boundary conditions (PBCs) is often employed.

In PBCs, the system is replicated in virtual space in one, two or three dimensions. In Fig. 2-1, system, indicated by a box, is replicated in two dimensions. Around the highlighted particle, a circle with radius R_c (the cutoff distance) can be seen.

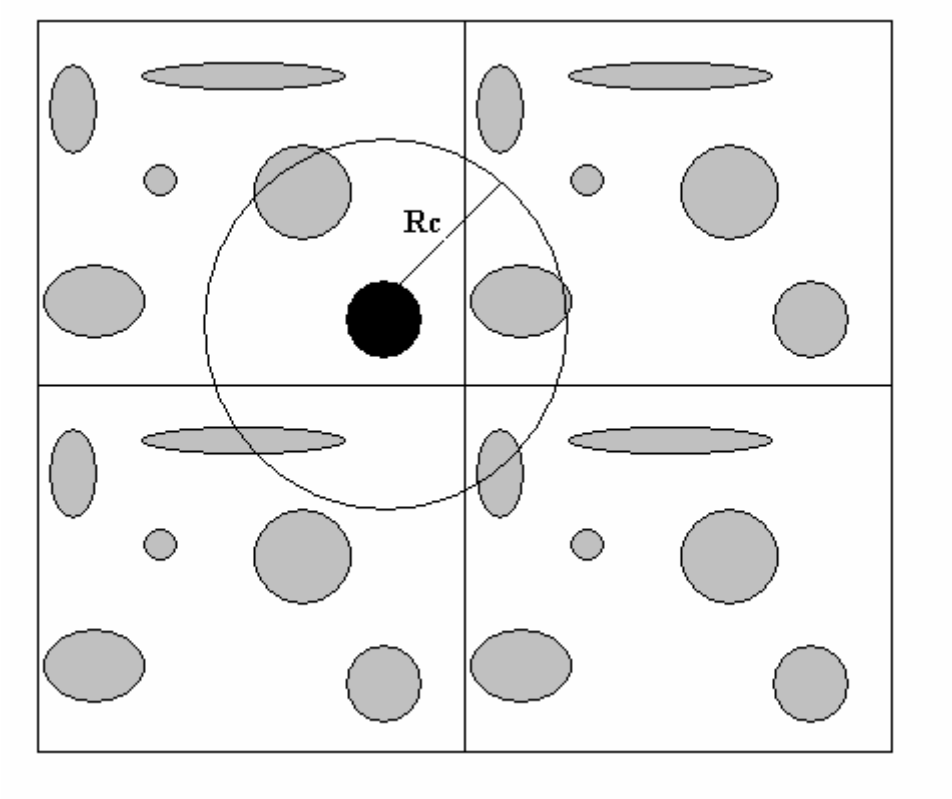


Figure 2-1: Schematic of periodic boundary conditions.[74]

Particles falling within the circles are neighbors of the highlighted particle. Beyond this distance the interactions are small enough to be neglected. If particle i is located at position vector **r** in the system, then an identical particle is assumed to be located at **r** + (i**x** + j**y** + k**z**), where i, j, and k are integers ranging from -∞ to +∞. However, if some particles are too far from other particles in the same box to interact, they may be very close to these particles in an adjacent box. Care is therefore needed to select a large enough system to avoid any interactions between the particle and its periodic images. In general, very large systems can be simulated with much smaller systems through the use of PBCs and macroscopic behavior of the system can consequently be predicted in a realistic manner.

Fig. 2-2 shows a schematic representation of the general procedure followed in MD algorithms. In order to achieve better results, many useful tools at each step have been

developed since the introduction of MD. Full details about each step of this algorithm are discussed below.

2.1.2 The MD Algorithm

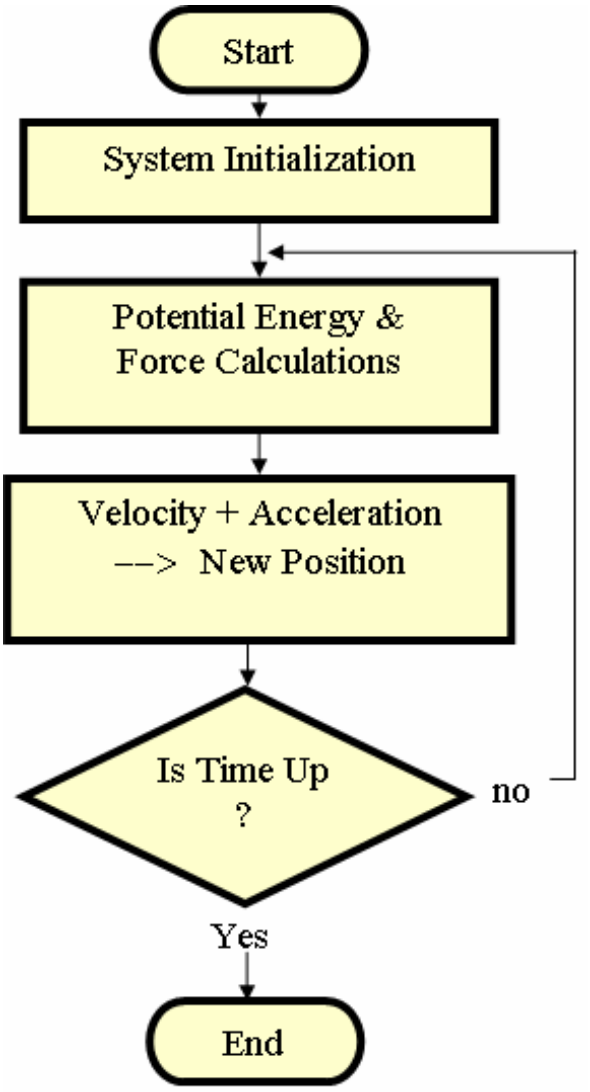


Figure 2-2: General flow chart of classical MD simulations. The loop keeps on repeating at every time step until total number of time steps are reached.

2.1.2.1 Initialization

When MD simulations start, certain information is provided in an input file, including the number of particles in the system, the number of time steps to be used in the simulation, the system temperature, the value of the time step to be used, the initial

coordinates of all the particles, and the initial velocities of all the particles. Based on the initial velocity of all the particles, the initial temperature is calculated based on Eq. 2-5. The initial velocities of all the particles are arbitrarily assigned depending on the need of certain particles to move at a certain kinetic energy to produce the desired initial system temperature.

2.1.2.2 Calculation of potential energy and force; Energy minimization

The potential energy between two particles is a function of the distance between them and is dependant on the characteristics of the particles. From the principle of conservation of energy, the sum of the potential energy U and the kinetic energy KE (= $mv^2/2$) is constant as shown in Eq. 2-7

$$\frac{mv^2}{2} + U = \text{constant} \qquad (2\text{-}7)$$

where v = dr/dt

So Eq. 2-7 can be written as follows:

$$\frac{1}{2} m \left(\frac{dr}{dt}\right)^2 + U = \text{constant} \qquad (2\text{-}8)$$

If Eq. 2-8 is derived with respect to time t, Eq.2-9 is derived.

$$\frac{d}{dt}\left[\frac{1}{2} m\left(\frac{dr}{dt}\right)^2 + U\right] = 0 \qquad (2\text{-}9)$$

$$m \left(\frac{dr}{dt}\right) \frac{d}{dt}\left(\frac{dr}{dt}\right) + \frac{dU}{dt} = 0 \qquad (2\text{-}10)$$

Potential energy is a function of position so Eq 2-10 can be rearranged as follows:

$$m \left(\frac{dr}{dt}\right) \frac{d}{dt}\left(\frac{dr}{dt}\right) + \frac{dU}{dr}\frac{dr}{dt} = 0 \qquad (2\text{-}11)$$

Also

$$a=\frac{dv}{dt}=\frac{d}{dt}\left(\frac{dr}{dt}\right)$$

Eq. 2-11 can be rewritten as:

$$ma = -\frac{dU}{dr} \text{ or } F_{ij,k} = -\left(\frac{\partial U_{ij}}{\partial r_{ij,k}}\right) \quad (2\text{-}12)$$

where k denotes the coordinate direction, F_{ij} is the force between particles i and j, and r_{ij} is the distance between particles i and j. Therefore, force is the negative derivative of potential energy with respect to the distance between two particles as shown in Eq. 2-13. The forces on particles i and j due to the potential energy between them is half of F_{ij} but they act in opposite directions. Consequently, the total sum of forces in the system is zero:

$$\sum_i \sum_{i<j} F_{ij,k} = -\sum_i \sum_{i<j} \left(\frac{\partial U_{ij}}{\partial r_{ij,k}}\right) \quad (2\text{-}13)$$

2.1.2.3 Integrator; Velocity Verlet

The most widely used method of integrating the equation of motion is integrator (Verlet 1967). This method provides a direct, numerical solution of Newton's second law F = ma. It is an iterative, numerical method in which, acceleration is calculated from the force acting on the atoms in the system and value of velocity is reflected from corresponding temperature of the system. With this method, the handling of velocities is a concern and introduces numerical imprecision. A Verlet equivalent algorithm was introduced by Swope, Anderson, Berens, and Wilson [75] called velocity Verlet, which is very effective in storing position, velocities, and acceleration all at the same time t with minimizing round off errors. Following equations are the calculation of position and velocity of the atoms after time δt depending on the value of acceleration and velocity at time t.

$$r(t + \delta t) = r(t) + \delta t\, v(t) + \tfrac{1}{2}\, \delta t^2\, a(t) \quad (2\text{-}14)$$

$$v(t + \delta t) = v(t) + \tfrac{1}{2}\, \delta t\, [a(t) + a(t + \delta t)] \qquad (2\text{-}15)$$

This method is simple to code, numerically stable, and convenient to use. It conserves forces and is guaranteed to conserve linear momentum. This method is excellent in energy conserving properties even at long time steps.[68]

2.1.2.4 Ewald summation

Ewald summation is a method for effectively summing the interactions between an ion and all its periodic images. It was originally developed by Ewald[76] and Madelung[77] for the study of ionic crystals. The Ewald method is most efficient when the medium in which the particles reside is conducting. Throughout the simulations, the distribution of charge in the central cell constitutes the neutral lattice unit cell that extends throughout space. In this method, each point charge is surrounded by a charge distribution of equal magnitude with opposite sign that spreads radially from the central charge. This extra distribution is called the screening distribution, which screens the interaction between the neighboring charges.

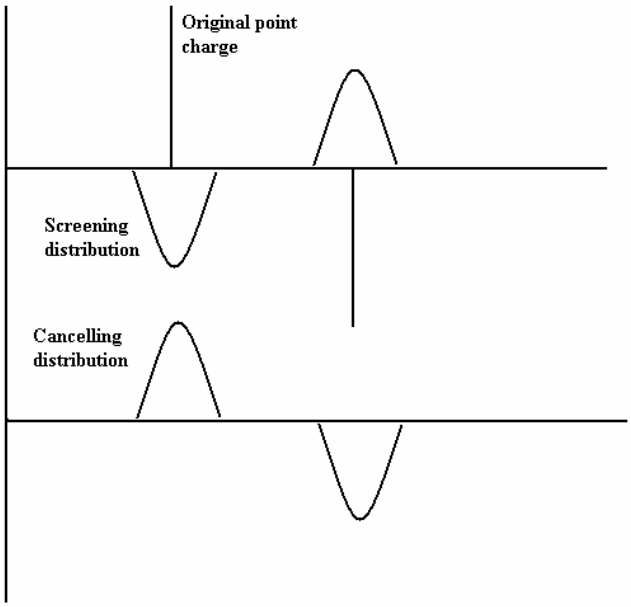


Figure 2-3: Charge distribution in Ewald summation[68]

The canceling distribution, which is also added to the calculations and is shown in Fig. 2-3, has the same sign as the original charge and the same shape as the screening distribution. The canceling distribution is summed in reciprocal space. Fourier transforms of the canceling distribution of each charge is added and the total is transformed back to real space. One more correction term, the self term, is used to cancel the distribution centered at the point charge with itself, is also subtracted from the calculation. Detailed equations of the potential energy of ionic interaction and their negative derivatives are given in the Appendix A.

2.2 Surfactant and Surface Model Details

Here, C_{12}TAB (n-dodecyltrimethylammoniumbromide) surfactant is examined in MD simulations in an aqueous medium in the presence of negatively charged silica surface. C_{12}TAB consists of a hydrophobic 12-carbon chain and a trimethylammonium head group that is hydrophilic, which makes the surfactant as a whole amphiphilic. The CH_3, CH_2, and trimethylammonium molecules are treated as single units, or pseudoatoms, to increase the efficiency of the simulations. The intra-molecular interaction (E_{intra}) of the surfactant is represented as the sum of the following terms:[78]

$$E_{intra} = E_{stretch} + E_{bend} + E_{torsion} + E_{LJ} \quad (2\text{-}16)$$

The bond stretching term is given by:

$$E_{stretch} = \tfrac{1}{2}\, k_r\, (b_{ij} - b_0)^2 \quad (2\text{-}17)$$

where b0 is the equilibrium bond length (0.1539 nm), kr is the bond force constant (9.5953 eV/Å^2). [78]

The bending potential term in Eq. 2-15 is given by:

$$E_{bend} = \tfrac{1}{2}\, k_b\, (\cos\theta_0 + (b_j{+}1 * b_i)/(b_j{+}1)(b_i)) \quad (2\text{-}18)$$

where $b_i = r_{i+1} - r_i$, r_i is the distance between two atoms, θ_0, has a value of 112.15°[78] and the force constant, k_b, is 1.3485 eV.[78] The contribution from torsion in Eq. 2-16 is given by:

$$E_{torsion} = kt\,(1-\xi_i)\,(\xi i^2 + a_1\xi_i + a_0) \qquad (2\text{-}19)$$

where $\xi_i = b_i - 1\ b_i + 1/b_0^2$, $b_i = r_{i+1} - r_i$, and $b_0 = 0.1539$ nm. The torsional force constant, kt, is equal to 0.4947 eV, the constant a_0 is equal to 0.1115, and the constant a_1 is equal to 0.6669.[78] Finally, the last term in Eq. 2-16, E_{LJ}, is given by a Lennard-Jones (LJ) potential where pseudo atoms are separated by at least two other pseudo atoms along the chain. The same term is used to model the interaction between pseudo atoms from different surfactants. This potential has the following form:

$$E_{LJ} = [6\,\varepsilon_{ij}/(n-6)]\,[(\sigma_{ij}/r_{ij})^n - (n/6)\,(\sigma_{ij}/r_{ij})^6] \qquad (2\text{-}20)$$

where n is 9 and the value of ε_{ij} and σ_{ij} are 4.35×10^{-3}eV and 4 Å, respectively.[78] The charge on the head group is +1.0 and the charge on the counter ion bromide is -1.0.

The potential used to model the water molecules is a simple point charge model (SPC). The intermolecular interactions between two water molecules consist of a Lennard-Jones potential between two oxygen atoms (in different water molecules) and a Coulombic electrostatic potential (charges are +0.41e on hydrogen and -0.82e on oxygen),[79] where e is the charge of an electron.

The LJ potential for water is:

$$\triangle\varepsilon_{LJ} = \sum_i \sum_j ((1/4\pi\varepsilon_\circ)q_iq_je^2/r_{ij} + B_{ij}/(r_{ij})^{12} - A_{ij}/(r_{ij})^6) \qquad (2\text{-}21)$$

where A = 27.14 eV Å^6 and B = 27315.25 eV Å^{12}.[79] The harmonic intra-molecular potential for oxygen and hydrogen within a water molecule is give by:

$$V_{intra} = \tfrac{1}{2}\,a\,[(\Delta r_1)^2 + (\Delta r_2)^2] + \tfrac{1}{2}\,b\,(\Delta r_3)^2 + c\,(\Delta r_1 + \Delta r_2)\,\Delta r_3 + d\,\Delta r_1\,\Delta r_2 \qquad (2\text{-}22)$$

where Δr_1 and Δr_2 are the stretches in the two O-H bond lengths and Δr_3 is the stretch in the H-H length. The values of a, b, c, and d are 58.24, 14.251, -9.1698, and 4.844 eV/Å^2, respectively.[79]

The interaction between the surfactant head groups and the water molecules is given by the following Lennard-Jones potential:

$$\triangle\varepsilon_{ab} = \sum_i \sum_j ((1/4\pi\varepsilon_\circ)q_i q_j e^2/r_{ij} + A_{ij}/r_{ij}^{12} - C_{ij}/r_{ij}^{6}) \qquad (2\text{-}23)$$

where $A_{ij} = (A_{ii}\ A_{jj})^{1/2}$, $A_{ii} = 4\ \varepsilon_i\sigma_i^{12}$, $C_{ij} = (C_{ii}\ C_{jj})^{1/2}$, $C_{ii} = 4\ \varepsilon_i\sigma_i^{6}$, and $\varepsilon_\circ$ is permittivity. The parameters for the LJ potentials are given in Table 2-2. [80]

Table 2-2: Parameters for the LJ potential used to model interactions between surfactant head groups, water molecules, and SiO_2.[80]

	$(CH_3)_3N^+$	Oxygen	Hydrogen	Si of SiO_2	O of SiO_2
Q(e)	0.25	-0.82	0.41	0.31	- 0.71
A_{ii} (eV Å^{12})	374358.54	27315.25	0.0	197319.2193	26051.5852
C_{ii} (eV Å^6)	97.07	27.0	0.0	66.0539	26.4645

The silica surface, which has an amorphous structure, is held fixed throughout the simulations. The Si ions in the surface each have a fixed positive charge of 0.31 and the O ions each have a fixed negative charge of 0.71. Thus the surface as a whole has a net negative charge. Figures 2-4 and 2-5 show the top and side view of silica surface used in the simulations.

The graphite surface is held fixed throughout the simulation. The flat graphite surface is hydrophobic and nonionic. Figures 2-6 and 2-7 show the top and side view of graphite surface used in the simulations.

The interaction between the surfactant head groups, CH_3/ CH_2, and water molecules with graphite surface is given by the following Lennard-Jones potential:

$$\triangle\varepsilon_{ab} = \sum_i \sum_j E_0/(n\text{-}m)[m(r_0/r_{ij})^n - n(r_0/r_{ij})^m] \quad (2\text{-}24)$$

where m= 3 and n= 9. Following table lists the parameters r_0 and E_0 for graphite-site interaction.

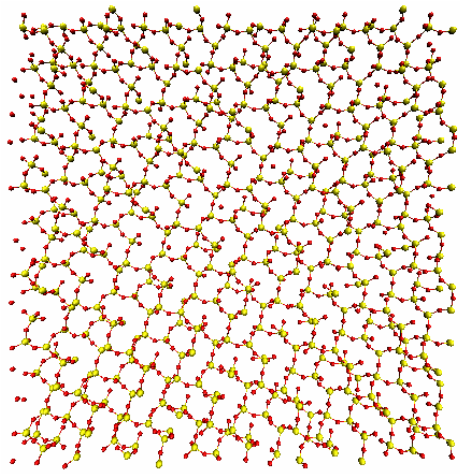


Figure 2-4: Silica surface (top view)

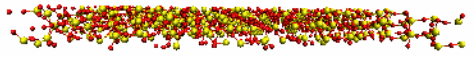


Figure 2-5: Silica surface (side view)

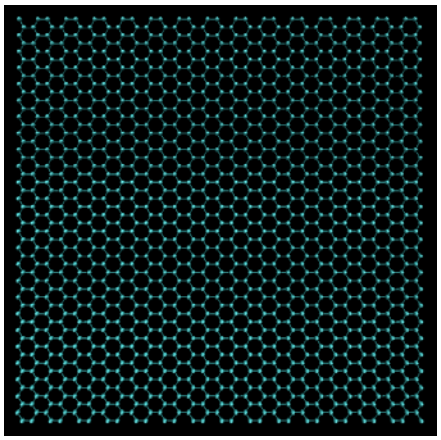


Figure 2-6: Graphite surface (top view)

Figure 2-7: Graphite surface (side view)

Table 2-3: 9-3 Lennard-Jones potential parameters for graphite-site interaction[65]

	$(CH_3)_3N^+$	Oxygen	Hydrogen	CH_3/ CH_2
r_0 (Å)	2.516	2.474	0.0	2.856
E_0 (eV)	0.02098	0.02006	0.0	0.0156

The potential used to model the long-range interactions between CH_2 and oxygen in the water molecules is:

$$\triangle\varepsilon_{ab} = \sum_i \sum_j (A_{ij}/r_{ij}^{12} - C_{ij}/r_{ij}^{6}) \qquad (2\text{-}25)$$

where $A_{ij} = (A_{ii}\ A_{jj})^{1/2}$, $A_{ii} = 4\ \varepsilon_i\sigma_i^{12}$ and $C_{ij} = (C_{ii}\ C_{jj})^{1/2}$ where $C_{ii} = 4\ \varepsilon_i\sigma_i^{6}$ [78,79]. Parameters for these equations are given in Table 2-4.

Table 2-4: Parameters for the LJ potential to model interactions between surfactant CH_2 groups and water molecules. [78,79]

	CH_2	Oxygen	Hydrogen
A_{ii}(eV Å^{12})	219874.14	27315.25	0.0
C_{ii} (eV Å^6)	59.67	27.0	0.0

Each simulation trajectory is allowed to run for a few tens of nanoseconds and the time step used is 1 fs. All the simulations are carried out in aqueous environments or at water-solid surface interfaces with surfactant concentrations above or below the cmc where aggregation of the surfactant is expected. Every simulation runs for a few hundred picoseconds to few tens of nanoseconds. Although these times scales are too short to entirely understand the true dynamic properties of micelles and surfactants (changes and fluctuations in the shape of micelles and micelle aggregation number can vary from 10^{-11} to 10^4 seconds), the simulations can provide dynamical information about the surfactant structures which is representative of the processes occurring in these systems.

2.3 Parallelization

We collaborated with Dr. Fortes and Mayank Jain from electrical engineering department to parallelize the code. The parallelization process deals with the following:

- Identifying the tasks that can be done in parallel
- Partitioning those tasks among processors

- Implementing a communication mechanism
- Providing processor synchronization during execution
- Integrating the parallel results

The main goal of parallel algorithms is to provide performance enhancements over sequential algorithms. This can be measured by the *speedup (p)* for *p* processors given by:

$$speedup(p) = \frac{time(1)}{time(p)} \tag{2-26}$$

As can be seen from Figure (Fig. 2-8), parallelization shows good performance up to 8 processors.

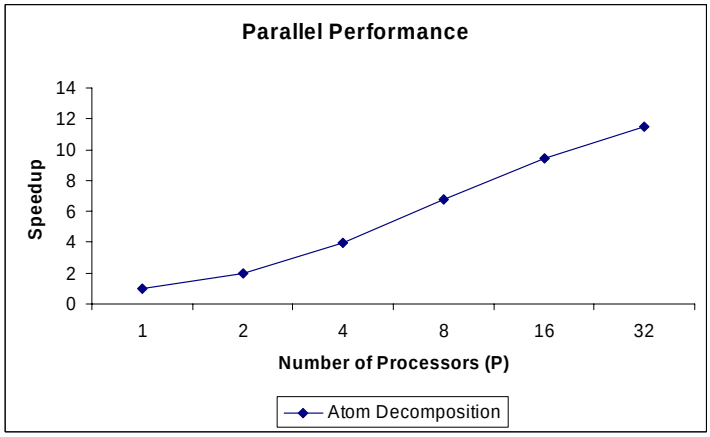


Figure 2-8: Parallel performance curve

MPICH-1.2 libraries were implemented for MPI, the standard for message-passing libraries. The program has been parallelized with the atom decomposition method.[81] In this method, each processor is assigned a subset of atoms and it updates their velocities and positions for the duration of the simulation regardless of where the atoms move in the physical domain. Each processor owns an identical copy of the entire system, i.e., it has the coordinates of all the particles at all times. The parallelization efforts resulted in good

scaling and made it possible for us to run large and realistic simulations. Parallelization techniques reduce the original sequential computation complexity of $O(n^2)$ to $O(n/p)$ for AD method, where n is the number of atoms in the system and p is number of processors used. The communication overheads are $O(np)$.

CHAPTER 3
C_{12}TAB SURFACTANTS/MICELLES IN AQUEOUS MEDIA

3.1 Spherical Micelle in Aqueous Media

To model the micelle structure of C_{12}TAB surfactants in water, 48 surfactants are initially placed such that they are close to each other in a spherical fashion in an aqueous medium of 4077 water molecules. The concentration of surfactants in the system is 0.23 M. The cmc of C_{12}TAB surfactants in aqueous media is 16 mM. Thus, the concentration is much above the cmc and spherical micelles are expected. The three-dimensional periodic boundary conditions extend 70 Å in each direction and are used to mimic an infinite aqueous medium around the initial surfactant micelle. The temperature of the system is maintained at 300 K by application of the velocity rescaling method to all the atoms in the system.

The micelle structure is allowed to evolve in the MD simulations to a lower-energy structure under equilibrium conditions. The total simulation time was 1 nanosecond. Figs. 3-1 and 3-2 are snapshots of the relaxed dense micelle structure after 0.3 and 0.9 nanoseconds respectively. Cationic and hydrophilic head groups are outside the structure shielding the hydrophobic chains inside the structure, while the surfactant tails are densely packed in the micelle interior. The aggregate is thus a densely packed structure. No surfactants separate from the main structure and no water molecules find their way inside the micelle interior over the course of the simulation. Interestingly, the micelle structure keeps transforming its shape into spherical (Fig. 3-1) (comparable to experimental results [82]) and elliptical (Fig. 3-2) shapes, which is in agreement with the results of other computer

simulations. [20] Some layering is also predicted to occur, as shown in Figs. 3-3 & 3-4. This is due to the strong hydrophobic interactions between the CH_3/CH_2 surfactant tails in the aqueous media. In each figure, dark blue represents the head group $N^+(CH_3)_3$ and green represents the tail molecules CH_3/CH_2. The water molecules are not shown in these figures for clarity.

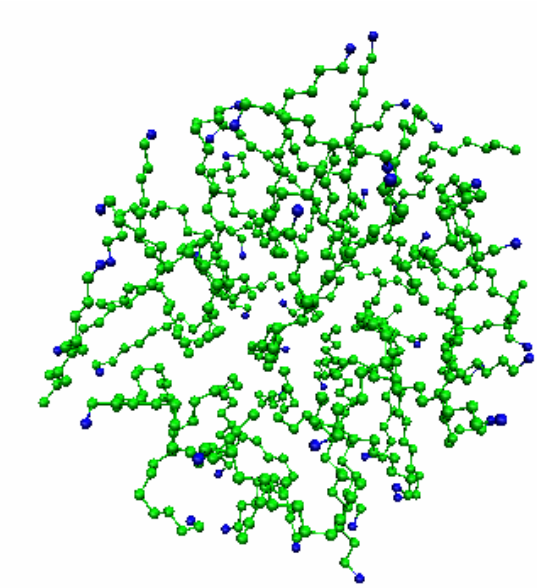


Figure 3-1: Snapshot of the micelle in aqueous medium after 0.3 nanoseconds of simulation run. The diameter of the micelle varies between 35 - 38 Å.

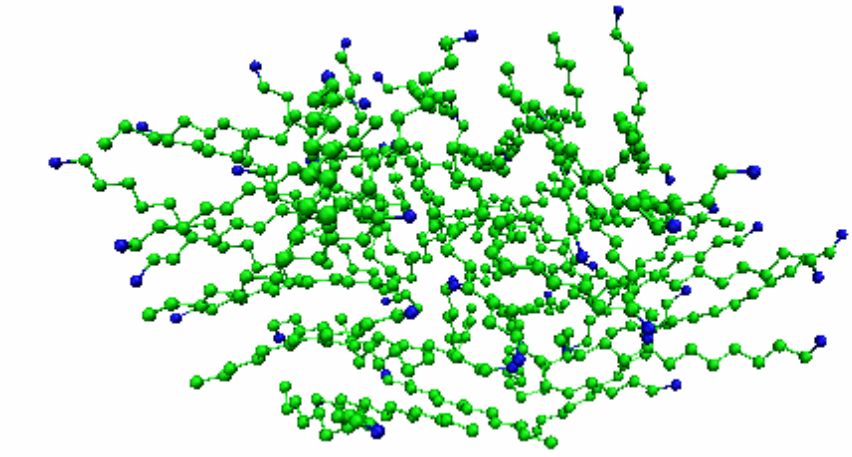


Figure 3-2: Snapshot of the micelle in aqueous medium after 0.9 nanoseconds. The diameter of the micelle varies between 28 – 65 Å.

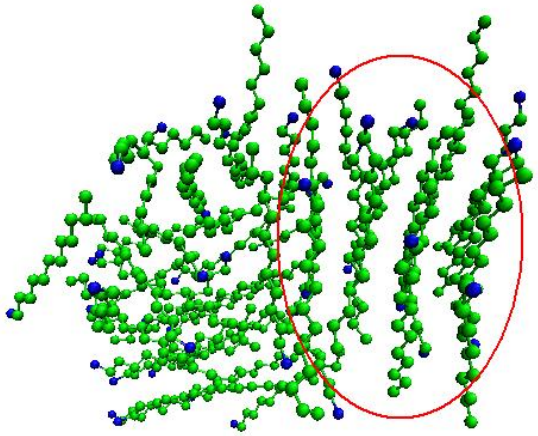


Figure 3-3: Snapshot of the micelle in aqueousmedium after 0.5 nanoseconds of simulation

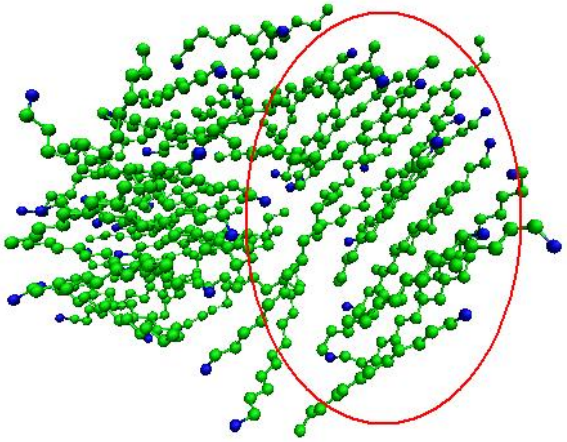


Figure 3-4: Snapshot of the micelle in aqueous medium after 0.6 nanoseconds of simulation.

Figs. 3-5 to 3-17 show stepwise evolution of spherical micelle in aqueous media.

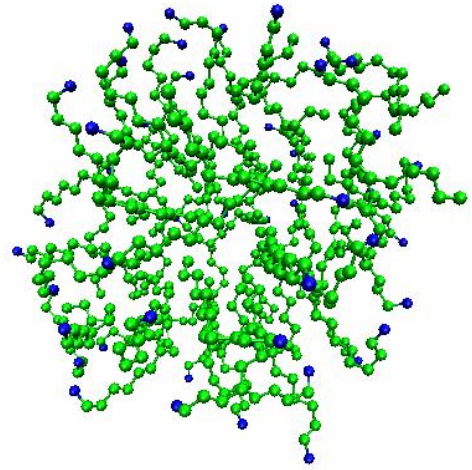


Figure 3-5: Initial Structure: 0.0ps-48 surfactants are relaxed in spherical geometry.

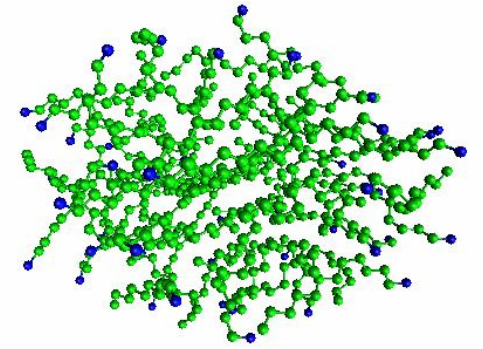


Figure 3-6: Structure at 30ps-Micelle keeps changing its structure towards a more favorable shape in which tails are coiled inside the micelle and head groups are on the surface shielding the tails from the aqueous media.

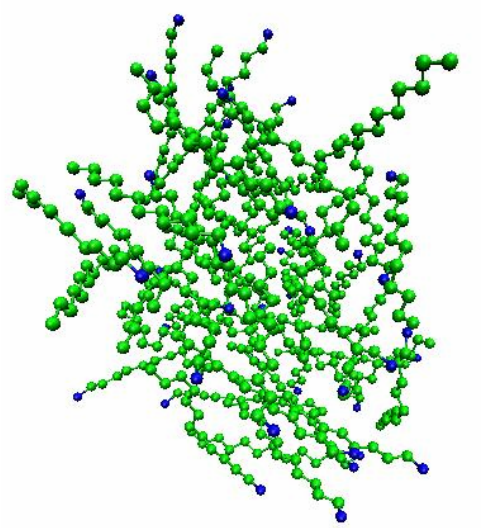


Figure 3-7: Structure at 100ps-Micelle keeps changing its structure towards a more favorable shape in which tails are coiled inside the micelle and head groups are on the surface shielding the tails from the aqueous media.

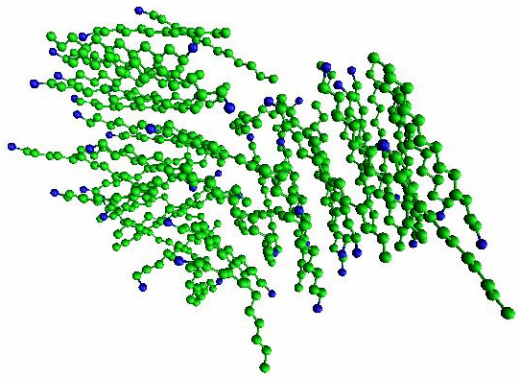


Figure 3-8: Structure at 150ps-Within the structure, layered surfactant arrangements are seen as shown in Fig. 3 and Fig. 4. This is due to the strong hydrophobic interactions between surfactant tails and the aqueous media.

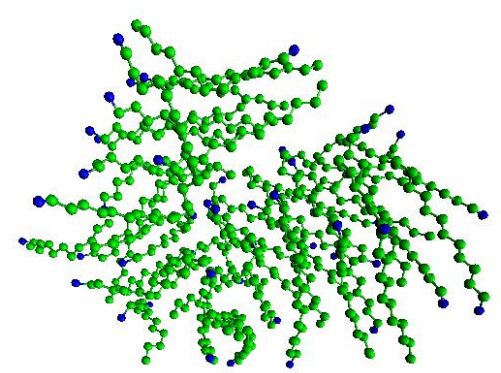


Figure 3-9: Structure at 200ps-Micelle structure forms a compact spherical shape.

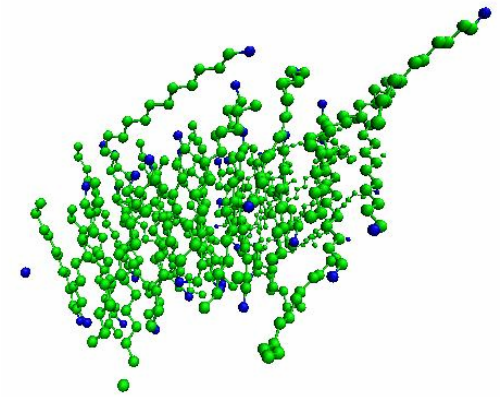


Figure 3-10: Structure at 300ps-Micelle structure appears to be compact with layered arrangement of surfactants.

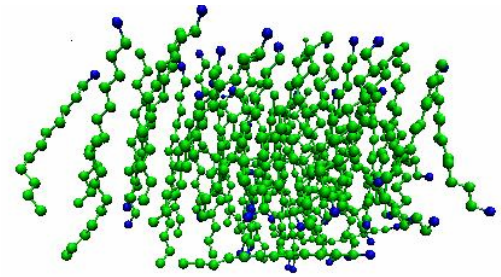


Figure 3-11: Structure at 450ps-Clear layered arrangement is visible.

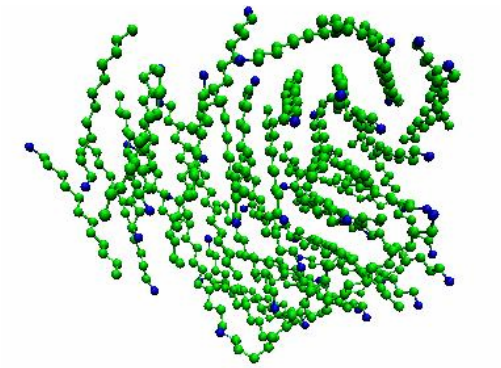


Figure 3-12: Structure at 500ps-Micelle structure evolves into spherical shape.

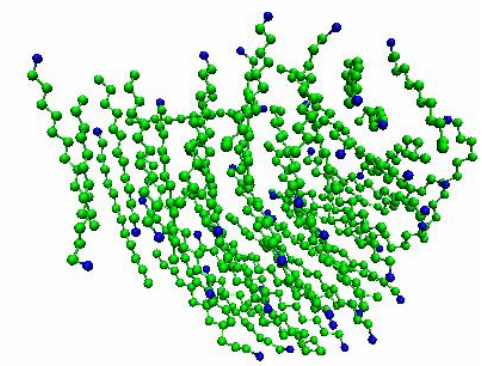


Figure 3-13: Structure at 600ps-Micelle structure appears to be spherical in shape with layered arrangements of surfactants.

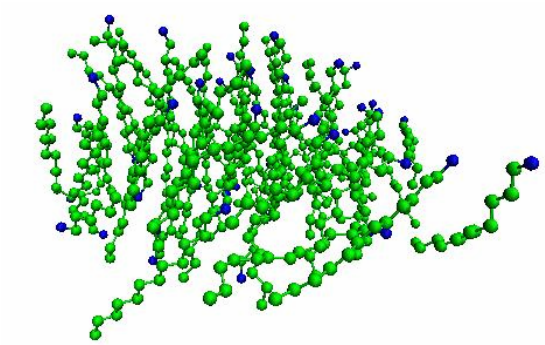


Figure 3-14: Structure at 700ps-One surfactant appears to move away from the micelle structure.

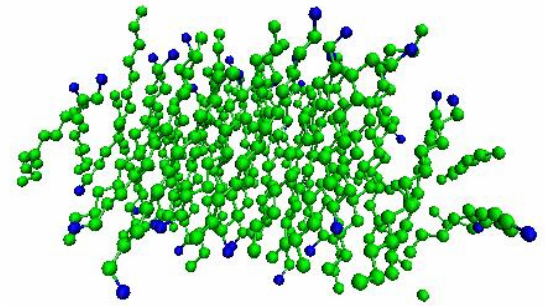


Figure 3-15: Structure at 800ps-Micelle structure appears elliptical in shape with head groups on the surface and tails coiled within the structure. All the surfactants are part of the micelle.

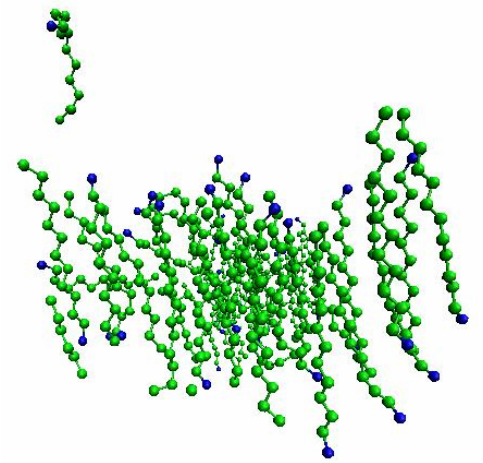


Figure 3-16: Structure at 950ps-Again one surfactant appears to move away from the structure. Surfactants form a layered arrangement.

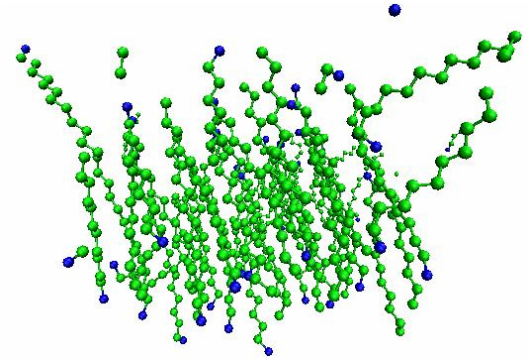


Figure 3-17: Structure at 1000ps-Structure forms spherical shape with layered arrangements of surfactants. All the surfactants are part of the structure.

3.2 Two Clusters in Aqueous Media

A single cluster of 24 surfactants was initially built in a spherical geometry. This cluster is replicated and thus two clusters are placed 10 Å apart in aqueous media. The other system details are similar to those discussed in section 3.1.

Two clusters are allowed to evolve in aqueous media in the MD simulations to a lower-energy configuration under equilibrium conditions. The total simulation time was 0.65 nanoseconds. Initially, exchange of surfactants occurs between the two clusters and eventually the two clusters approach each other and eventually merge into one another to form a single, larger micelle. The micelle thus formed has a spherical shape, with all the

head groups on the surface of the structure and the tails randomly arranged inside the structure. The aggregate is thus densely packed. No surfactants appear to move away from the micelle and no water molecules find their way inside the micelle interior over the course of the simulation. Thus the growth mechanism of micelle follows Smoluchowski model where cluster-cluster coalesce and form a bigger cluster. It agrees with the prediction, as the concentration of surfactant is much above cmc.[2,19] In each figure, two clusters of 24 surfactants are shown in different colors. In the left cluster, dark blue represents the head group $N^+(CH_3)_3$ and green represents the tail molecules CH_3/CH_2. In the right cluster, red represents the head group $N^+(CH_3)_3$ and yellow represents the tail molecules CH_3/CH_2.The water molecules are not shown in these figures for clarity.

Figs. 3-18 to 3-26 show stepwise evolution of two separate clusters of 24 surfactants each in aqueous media. Evolution of two small clusters in aqueous media is being tracked in the MD simulations

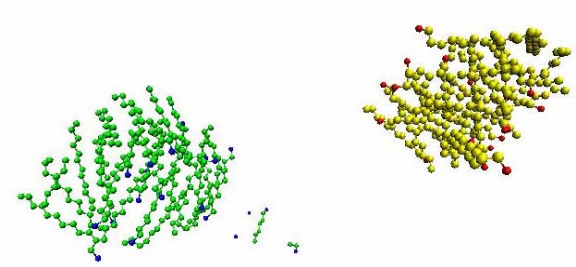


Figure 3-18: Initial Structure: 0.0ps-Two clusters of 24 surfactants each are placed 10 Å apart in aqueous media.

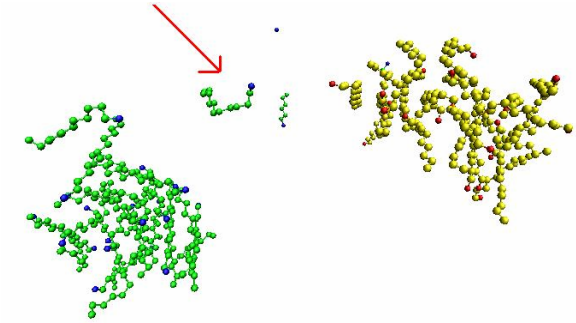


Figure 3-19: Structure at 195ps-The arrow indicates that one surfactant has detached from the left cluster and now appears in between two clusters.

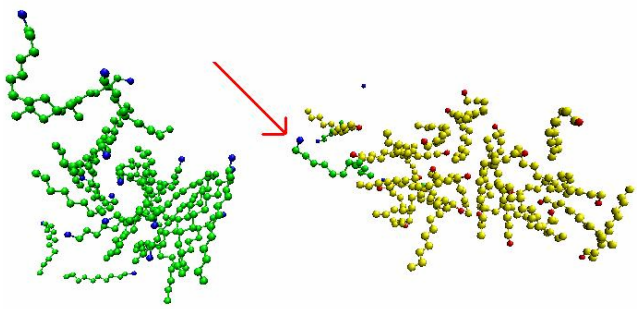


Figure 3-20: Structure at 210ps-The surfactant discussed above has attached itself to the right cluster. Exchange of surfactants between two clusters is evident.

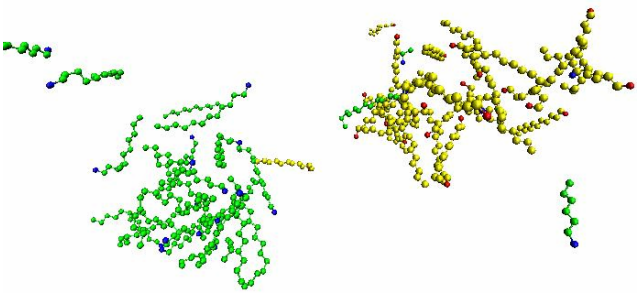


Figure 3-21: Structure at 295ps-Some of the surfactants appear to be away from their respective clusters. Both clusters appear to be away from each other. But both clusters have exchanged surfactants.

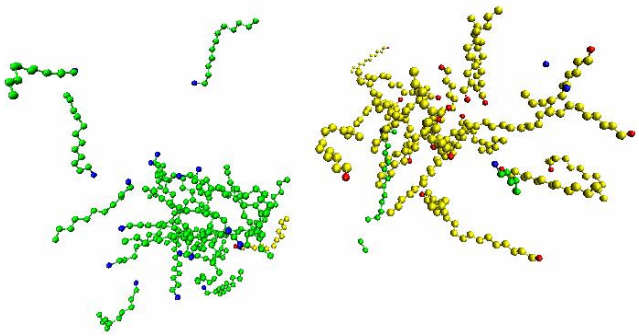


Figure 3-22: Structure at 330ps-Two clusters are coming closer to each other.

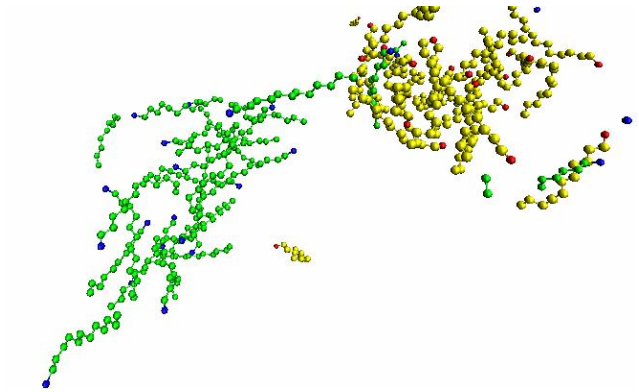


Figure 3-23: Structure at 375ps-Two clusters are attaching to each other.

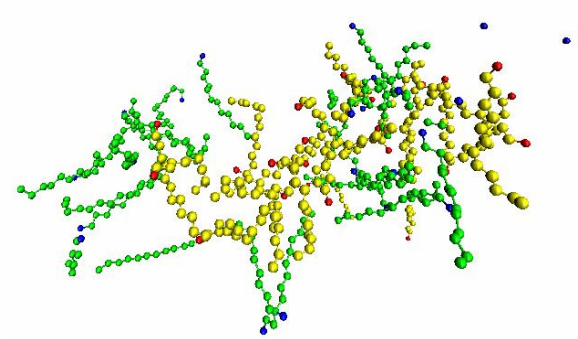


Figure 3-24: Structure at 510ps-Two clusters have attached to each other and are trying to form single micelle structure. Neither of the clusters stays together within itself. There is a complete intermixing of both the clusters.

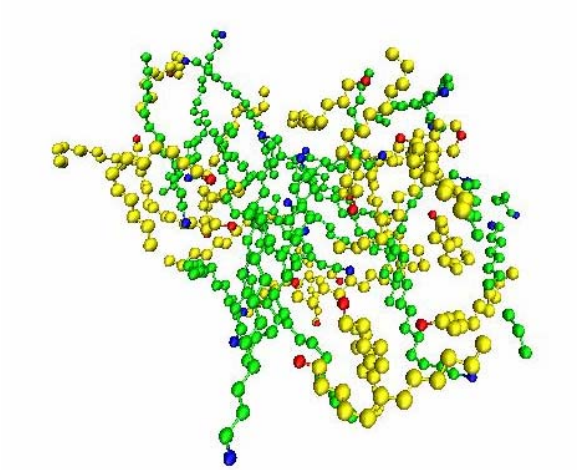


Figure 3-25: Structure at 600ps-Two clusters finally attached and formed single micelle structure.

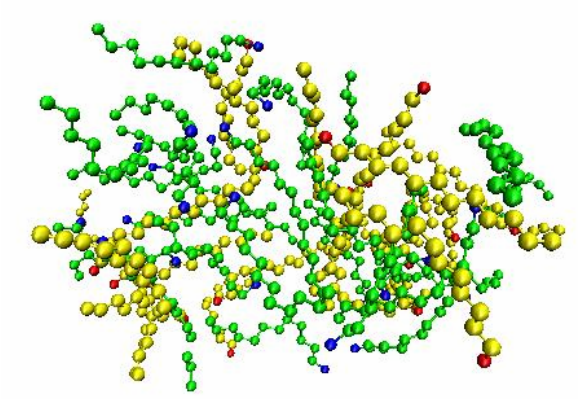


Figure 3-26: Structure at 650ps-Compact single micelle structure has formed.

3.3 Monolayer in Aqueous Media

A monolayer of 48 surfactants is initially built and placed in aqueous media. All other system details are the same as discussed in section 3.1. The monolayer structure is

allowed to evolve in the MD simulations to a lower-energy structure under equilibrium conditions. The total simulation time was 0.3 nanoseconds.

Head groups start repelling each other due to columbic repulsion. Tails start coming together due to hydrophobic attraction avoiding the water molecules around them. The structure starts swelling on the head group side and narrowing on tails side. Due to hydrophobic repulsion between tails and water molecules and hydrophilic attraction between head groups and water molecules, head groups start wrapping around the structure and start appearing on the other side of all the head groups. Tails of surfactant begin to randomize within the structure and eventually a spherical micelle results at the end of the simulation which can be seen in Figs. 3-33 and 3-34. The aggregate is thus a densely packed structure. No surfactants separate from the main structure and no water molecules find their way inside the micelle interior over the course of the simulation.

In each figure, dark blue represents the head group $N^{+}(CH_3)_3$ and green represents the tail molecules CH_3/CH_2. The water molecules are not shown in these figures for clarity.

Figs. 3-27 to 3-34 show stepwise evolution of monolayer of 48 surfactants in aqueous media. Evolution of monolayer is being tracked

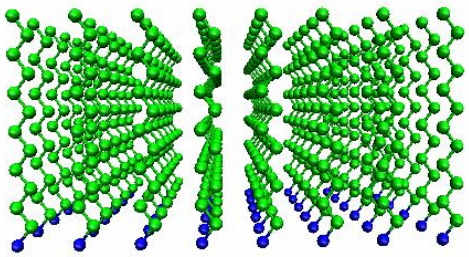


Figure 3-27: Initial Structure: 0.0ps-Monolayer of 48 surfactants is placed in aqueous media.

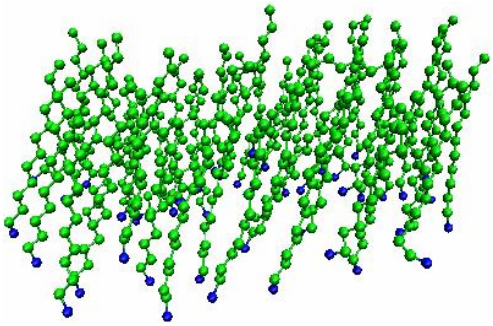


Figure 3-28: Structure at 11ps-All the tails come closer to each other due to hydrophobic interaction in the presence of water molecules. All the head groups repel each other due to columbic repulsion.

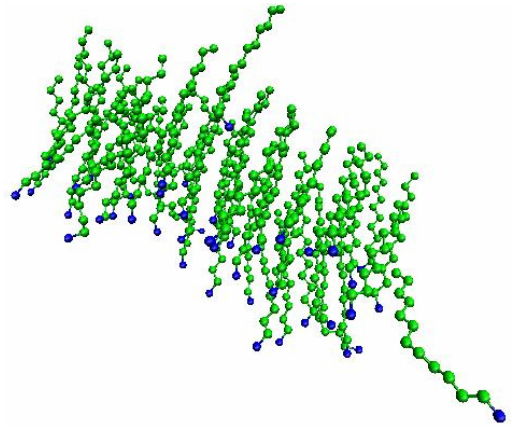


Figure 3-29: Structure at 50ps-Aggregate spreads from the head group side and narrows at the tail ends.

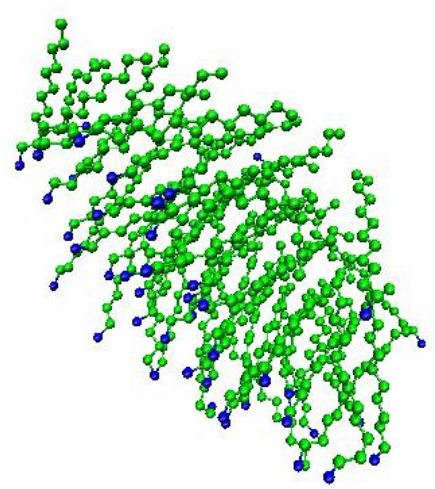


Figure 3-30: Structure at 100ps-Due to hydrophobic repulsion between tails and water molecules and hydrophilic attraction between head groups and water molecules, some head groups appear to be on the other side of all the head groups. They try to wrap around the structure to shield the tails from water molecules.

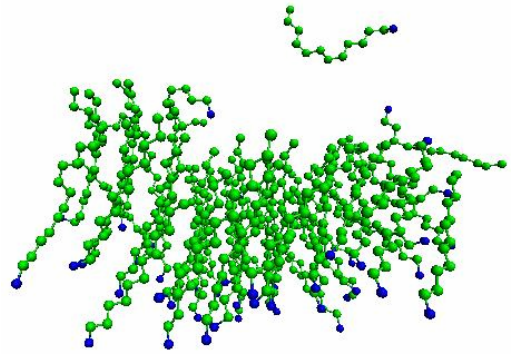


Figure 3-31: Structure at 150ps-One surfactant appears to be away from the structure. Some head groups appear to be at the other side of all the rest of the head groups.

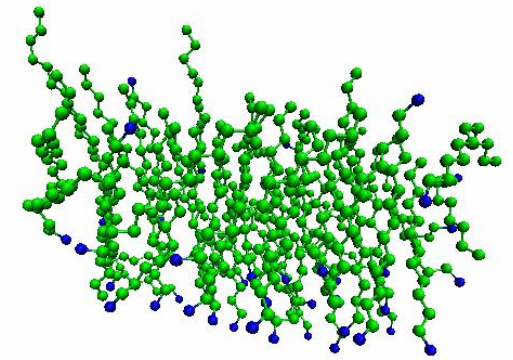


Figure 3-32: Structure at 200ps-Some more head groups appear to be at the other side.

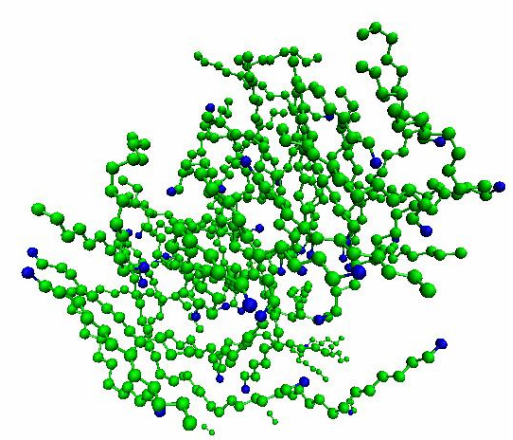


Figure 3-33: Structure at 250ps-Structure has randomized itself to form spherical micelle with head groups all around the surface of the structure and tails coiled inside.

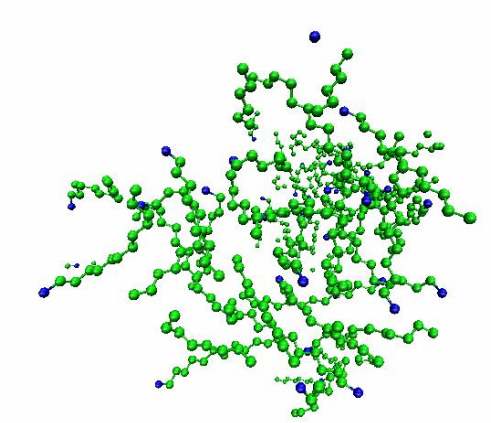


Figure 3-34: Structure at 300ps-Structure has evolved into spherical micelle.

3.4 Bilayer in Aqueous Media

Bilayer of 48 surfactants is initially built and placed in aqueous media. The other system details are the same as discussed in section 3.1. The bilayer structure is allowed to evolve in the MD simulations to a lower-energy structure under equilibrium conditions. The total simulation time was 1.3 nanoseconds.

Head groups start repelling each other due to columbic repulsion. Tails start coming together due to hydrophobic attraction in the presence of water molecules around them. The structure starts swelling from the head group side at both ends of bilayer. Due to hydrophobic repulsion between tails and water molecules and hydrophilic attraction between head groups and water molecules, head groups start wrapping around the structure and start appearing all around the surface of the structure. Tails of surfactant begin to randomize within the structure and eventually, a spherical micelle results at the end of the simulation which can be seen in Figs. 3-41 to 3-45. The aggregate is thus a densely packed structure. No surfactants separate from the main structure and no water molecules find their way inside the micelle interior over the course of the simulation.

In each figure, dark blue represents the head group $N^{+}(CH_3)_3$ and green represents the tail molecules CH_3/CH_2. The water molecules are not shown in these figures for clarity.

Figs. 3-35 to 3-45 show stepwise evolution of bilayer of 48 surfactants in aqueous media. Evolution of bilayer is being tracked

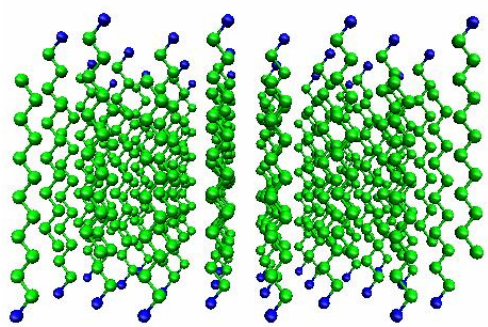


Figure 3-35: Initial Structure: 0.0ps-Bilayer of 48 surfactants is placed in aqueous media. Half of the head groups of the surfactants are pointed in the opposite direction to other half.

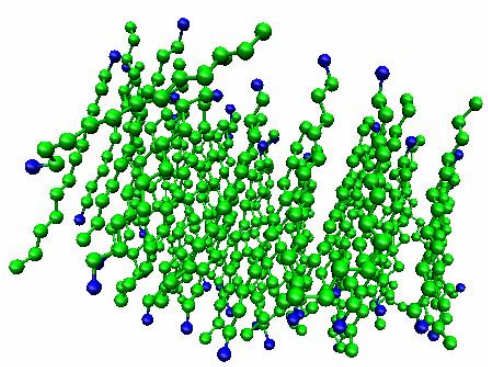


Figure 3-36: Structure at 155ps

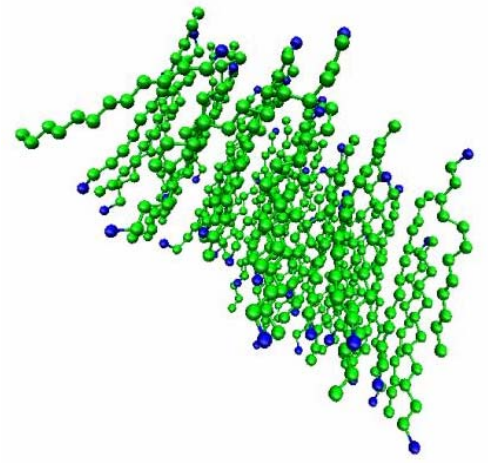


Figure 3-37: Structure at 350ps-Head groups repel each other due to columbic repulsion. Tails try to shield themselves from water molecules.

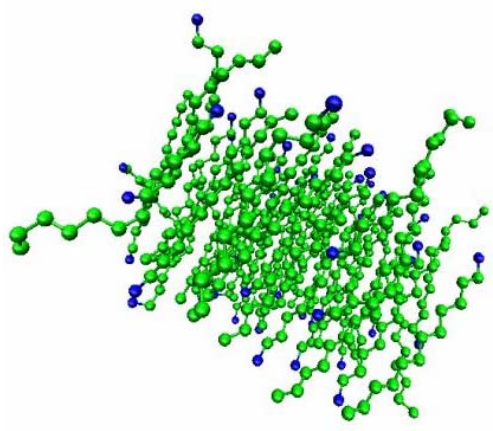


Figure 3-38: Structure at 435ps-Surfactant head groups try to wrap around themselves on the surface of the structure to shield tails from water molecules.

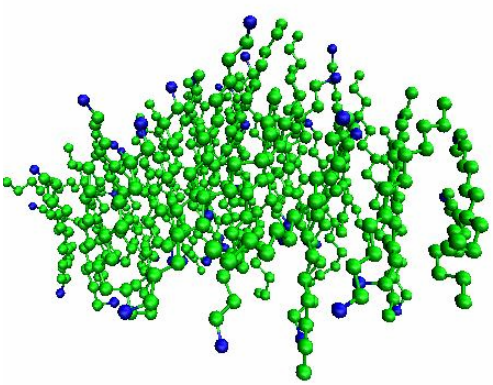


Figure 3-39: Structure at 645ps-Structure is beginning to curve on the surface, where head groups are on the surface and tails are coiled inside

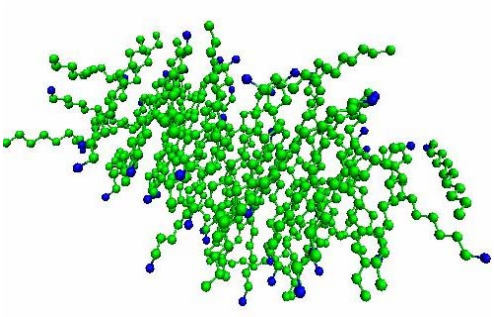


Figure 3-40: Structure at 970ps-Head groups are seen all around the structure.

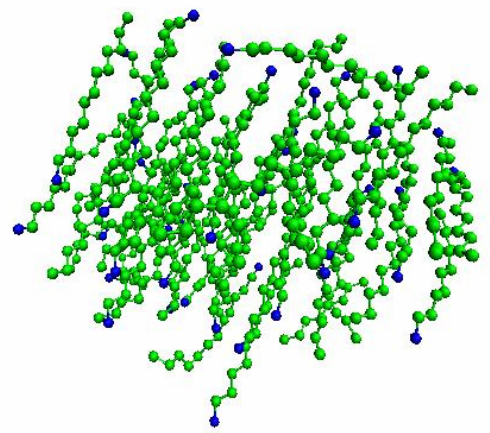


Figure 3-41: Structure at 985ps

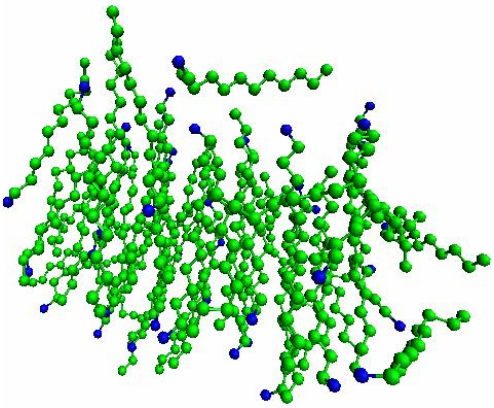


Figure 3-42: Structure at 995ps-Again, surfactant tails appear to be randomized within the structure. The head groups are all around the surface of the structure and the tails are coiled inside.

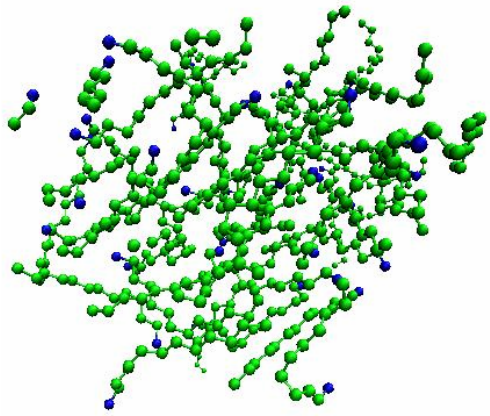


Figure 3-43: Structure at 1230ps-Structure is spherical with randomized tail arrangement and head groups are shielding tails from water molecules on the surface of the structure.

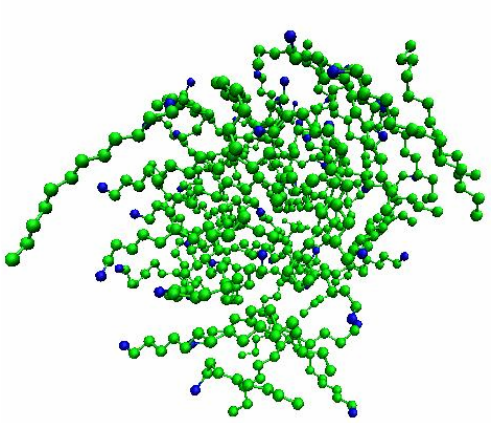


Figure 3-44: Structure at 1270ps-Structure is spherical with randomized tail arrangement and head groups are shielding tails from water molecules on the surface of the structure.

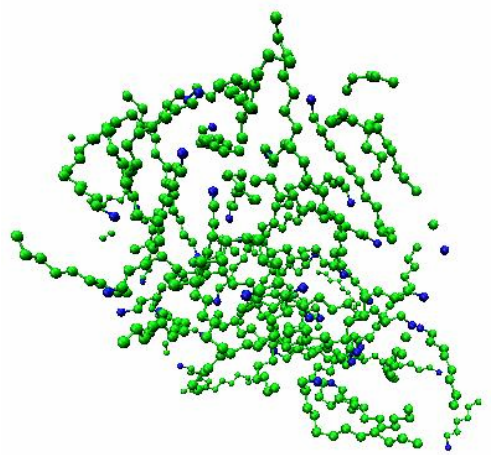


Figure 3-45: Structure at 1335ps-Structure is spherical with randomized tails arrangement and head groups are shielding tails from water molecules on the surface of the structure.

3.5 Summary

The emerging technologies such as controlled drug delivery, abrasives for precision polishing, coating, paints, and nano composite materials are increasingly relying on nano particulate material to achieve optimum performance. It is very important to produce well dispersed nano particulate dispersions. As self aggregate surfactant structures are optimally used as dispersants, the understanding of different structures in the bulk and on the dispersing particles is adequately needed. Due to limitations in the experimental techniques it is hard to achieve atomic/molecular level understanding of evolution of

micellar structure in the bulk. Computer simulation can be a helpful tool in providing this understanding at atomic/molecular level. In this study we are using MD simulation to study kinetics, equilibrium structure, and growth mechanism of micelle in the aqueous media.

The simulations indicate that self-aggregated surfactant structures form spherical or elliptical structures in bulk water at concentrations much above the surfactant cmc. Within the structure, some layering of surfactants is also predicted to occur, as shown in Figs. 3-3 & 3-4. This may be attributed to the strong hydrophobic interactions between the CH_3/CH_2 surfactant tails in the aqueous media. The simulations with two clusters show initial exchanges of surfactants between the clusters; eventually two clusters coalesce and form a single micelle. Thus it follows Smoluchowski model for growth of micelle when the concentration of surfactant is much above cmc[2,19]. In the simulation of monolayer in the bulk, the structure is far from equilibrium, where tails of surfactants are directly in contact with water molecules. Evolution shows that tails start rearranging themselves within the structure and eventually leads to spherical micelle having head groups on the surface of the sphere and tails coiled inside randomly. Similar results can be seen in case of bilayer in the bulk.Molecular level computer simulations such as those reported here complement experimental methods by providing molecular level insight into the evolution of surfactants and micelle in the bulk. This insight is expected to enhance the achievement of optimum performance in emerging and mature technologies such as controlled drug delivery, abrasives for precision polishing, coating, paints, and nano composite materials, which are increasingly relying on nano particulate materials which must be well-dispersed in solution with surfactants.

CHAPTER 4
C_{12}TAB SURFACTANTS/MICELLES AT SOLID-LIQUID INTERFACE

4.1 At the Silica/Liquid Interface

The following simulations were designed to study the discrete self aggregation of surfactants formed on silica surfaces and to better understand experimental data from AFM,[1,43,45] ellipsometry,[4] and NR[5-7] experiments. Results from these techniques indicate that C_{12}TAB form flat elliptical micelles on silica surfaces above cmc with little or no connecting necks between structures as shown in Fig. 4-1.

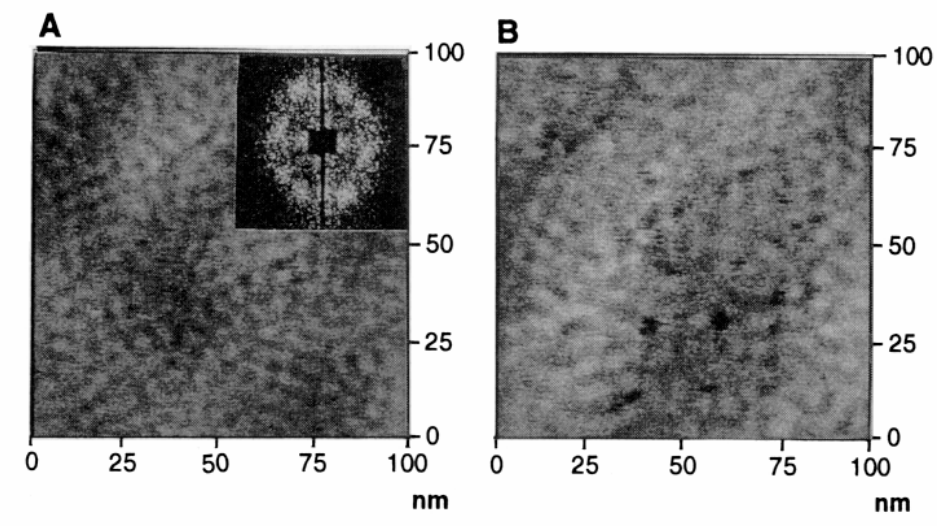


Figure 4-1: AFM images of aggregates of cationic C14TAB surfactants on silica (concentration =7 mM)[1]

4.1.1 Spherical Micelle on Silica

The simulations, shown in Fig. 4-2, were carried out with the 48-surfactant aggregate structure discussed in this section. The aggregate was placed 6 Å from the negatively charged silica surface with dimensions of 60 Å on each side within the plane of the surface and a slab thickness of 5 Å. The temperature of the system is maintained at

300 K by application of the velocity rescaling method to all the atoms in the system except the surface atoms which are held rigid.

The simulations predict that the round micelle adsorbs onto the silica surface without any connections to adjacent micelles through the applied periodic boundary conditions, which is consistent with experimental data. In the experimental results with AFM shown in Fig. 4-1, small dots are micelles adsorbed on silica and there is no connection between these dots. While in simulations, the adsorbed micelle does not extend at the side, which can be interpreted by no connection between adjacent micelles through applied periodic boundary conditions.

As the simulation evolves (Fig. 4-1), the structure flattens into an elliptical shape, as shown in Fig. 4-3, and the head groups are columbically attracted to the oppositely charged sites on the silica surface. In each figure, dark blue represents the head group $N^{+}(CH_3)_3$ and green represents the tail molecules CH_3/CH_2. Surface atoms (Si and O) are represented by yellow (Si) and red (O). The water molecules are not shown in these figures for clarity.

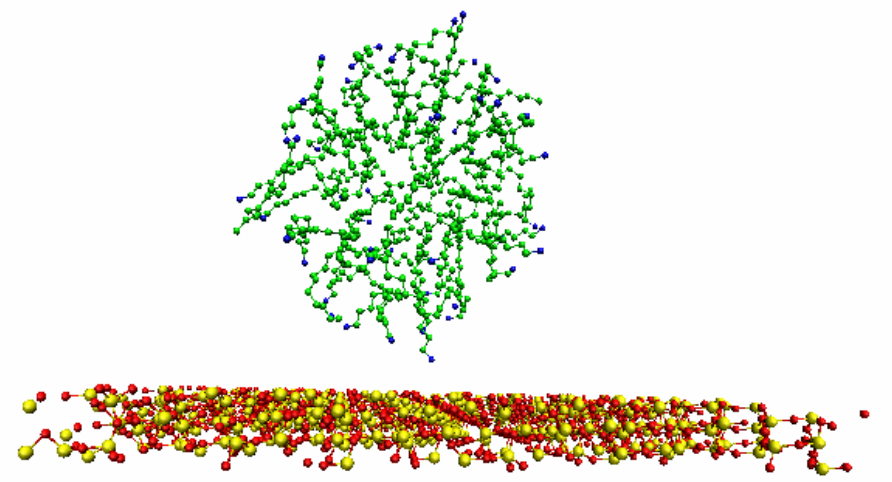


Figure 4-2: Snapshot of the initial configuration of the micelle on a silica substrate in aqueous medium. The dimensions of the silica layer are 60 Å by 60 Å by 5 Å

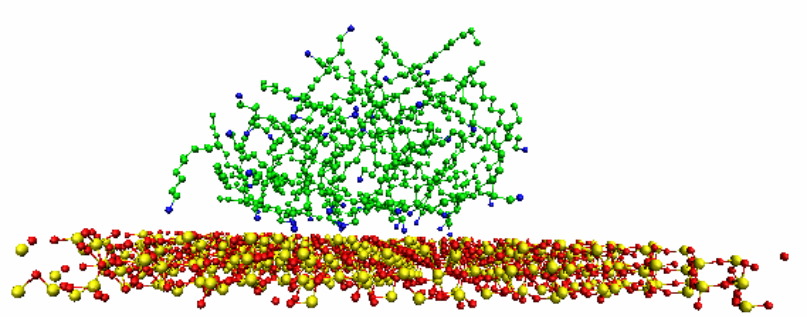


Figure 4-3: Snapshot of the system after 0.5 nanoseconds of MD simulation. The diameter of adsorbed micelle varies between 21 and 45 Å.

Figs. 4-4 to 4-18 show stepwise evolution of spherical micelle adsorption on hydrophilic negatively charged silica surface. Evolution of the spherical micelle structure on hydrophilic negatively charged silica is being tracked.

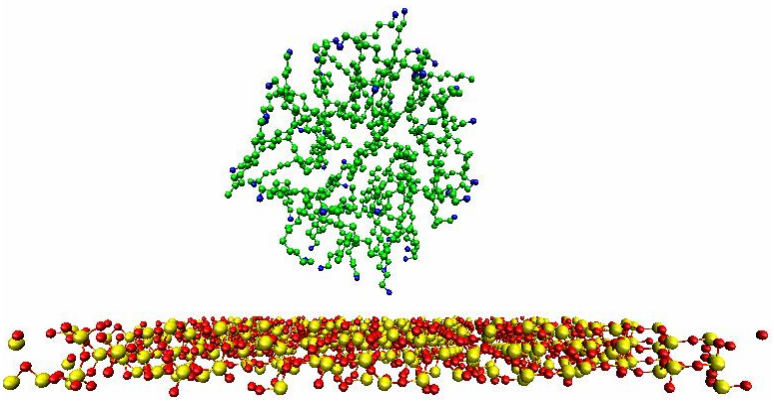


Figure 4-4: Initial structure at 0ps-Spherical micelle is placed 6 Å from the negatively charged silica surface as shown in Fig. 2.

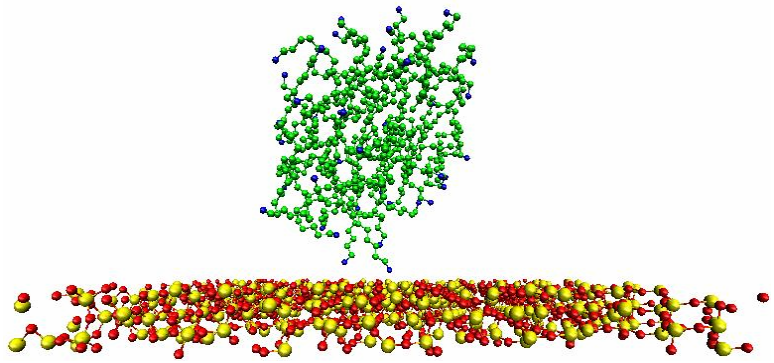


Figure 4-5: Structure at 20ps-Micelle structure begins to lower towards the surface due to columbic attraction between cationic head group of surfactants and negatively charged silica surface.

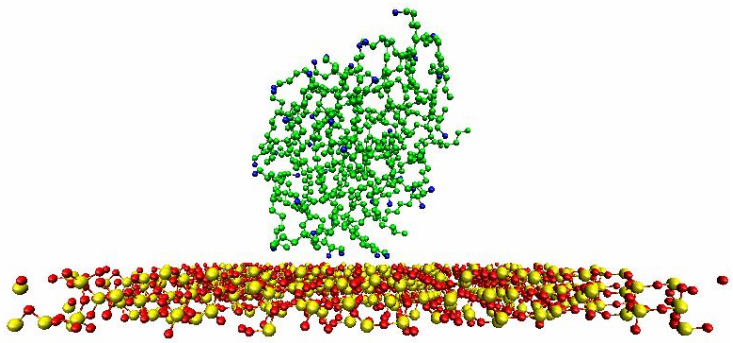


Figure 4-6: Structure at 30ps-Micelle structure adsorbs on the surface as multiple head groups of the surfactants adsorb on the oppositely charged silica surface.

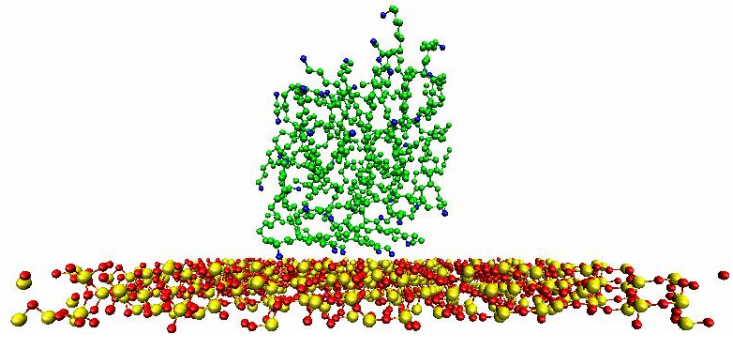


Figure 4-7: Structure at 40ps-Micelle continues to adsorb on the silica surface and flattens on the surface.

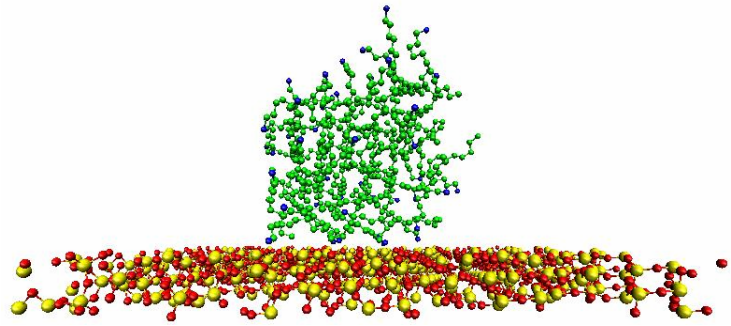


Figure 4-8: Structure at 50ps-Micelle structure stays adsorbed and keeps flattening on the silica surface.

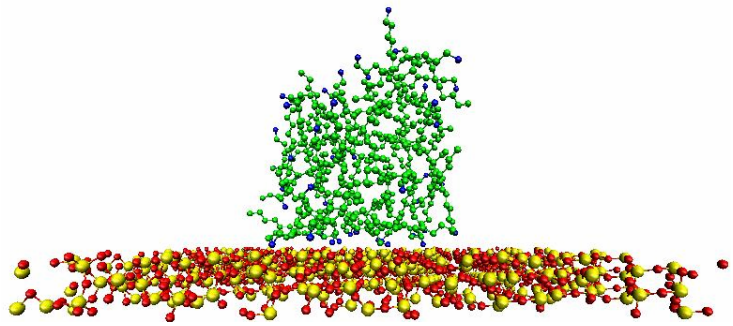


Figure 4-9: Structure at 65ps-Micelle structure starts spreading itself as more head groups try to adsorb on to silica surface.

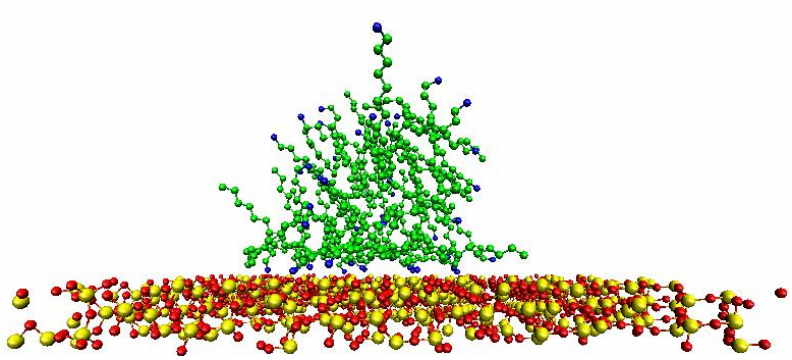


Figure 4-10: Structure at 100ps-Micelle structure keeps spreading itself onto silica surface.

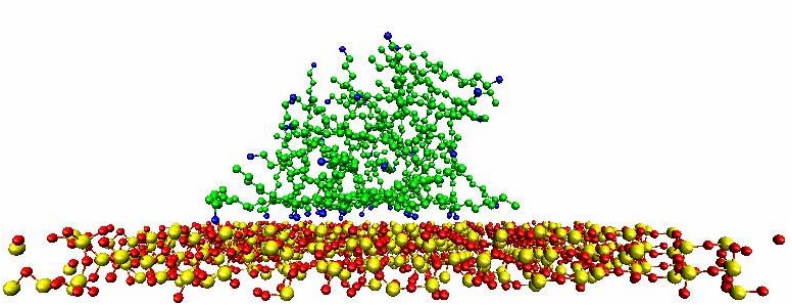


Figure 4-11: Structure at 115ps-Micelle stays adsorbed with flat elliptical shape.

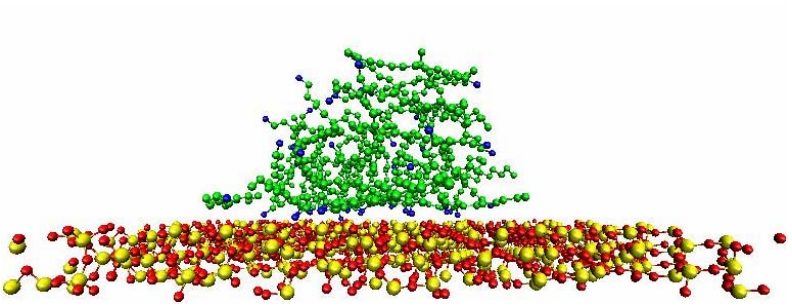


Figure 4-12: Structure at 150ps-Micelle stays adsorbed with flat elliptical shape.

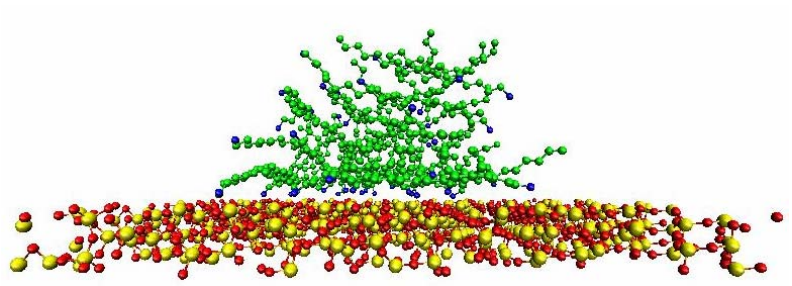


Figure 4-13: Structure at 200ps-Micelle stays adsorbed with flat elliptical shape.

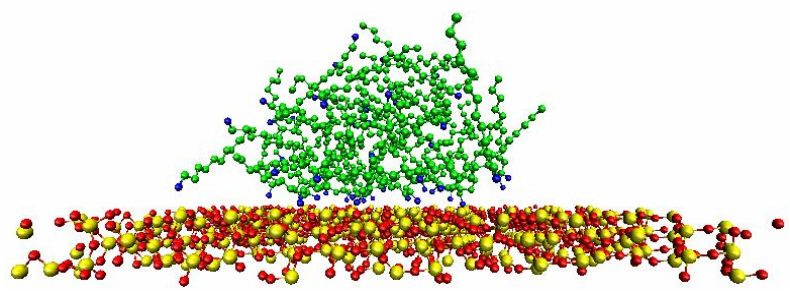


Figure 4-14: Structure at 223ps-One surfactant tries to come out of the structure and adsorb on to silica.

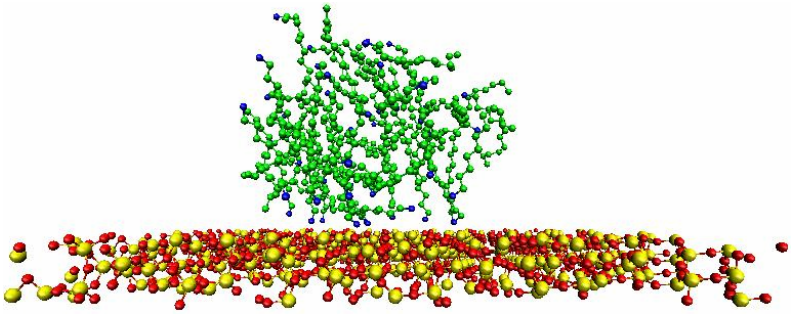


Figure 4-15: Structure at 245ps-Micelle structure reverts back in to flat elliptical shape as all the surfactants are part of the micelle.

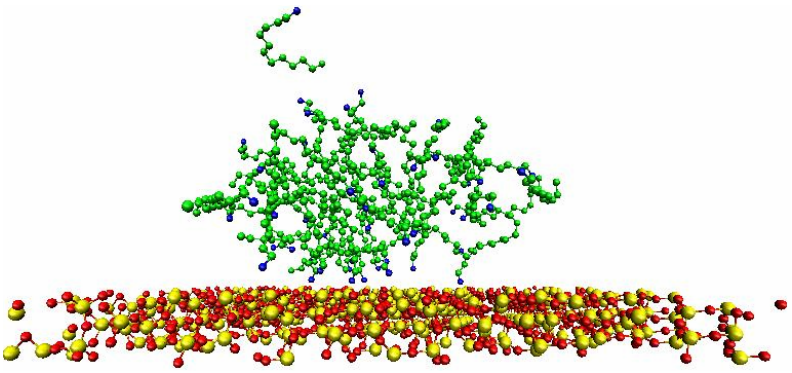


Figure 4-16: Structure at 262ps-One of the surfactants is seen to leaving from the micelle. The micelle seems to be more spread on the silica.

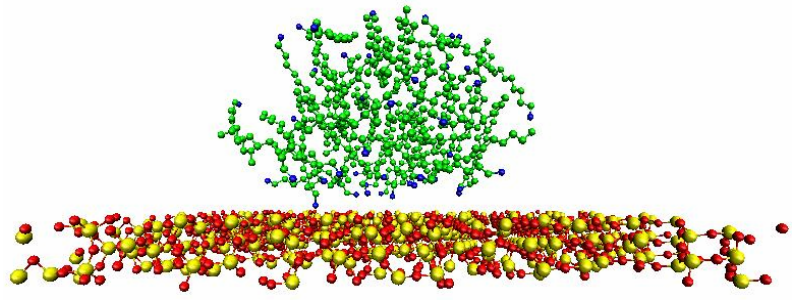


Figure 4-17: Structure at 300ps-All the surfactants are part of the micelle.

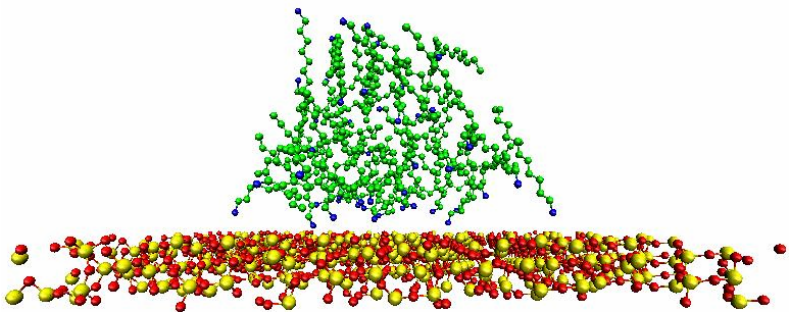


Figure 4-18: Structure at 335ps-Micelle stays adsorbed with flat elliptical shape.

4.1.2 Monolayer on Silica

The simulation, shown in Fig. 4-19 to 4-32, was carried out with the monolayer of 48-surfactant aggregate structure placed on hydrophilic negatively charged silica surface. The monolayer was placed 6 Å from the negatively charged silica surface with dimensions of 60 Å on each side within the plane of the surface and a slab thickness of 5 Å. The temperature of the system is maintained at 300 K by application of the velocity rescaling method to all the atoms in the system except the surface atoms which are held rigid.

The simulation predicts that the monolayer adsorbs on the silica, head groups start wrapping around the structure away from the surface and evolves in to flat elliptical micelle without any connections to adjacent micelles through the applied periodic boundary conditions. This result is consistent with experimental data. In the experimental results with AFM shown in Fig. 4-1, small dots are micelles adsorbed on silica and there is no connection between these dots. While in simulations, the adsorbed micelle does not extend at the side, which can be interpreted by no connection between adjacent micelles through applied periodic boundary conditions. The head groups are columbically attracted to the oppositely charged sites on the silica surface. In each figure, dark blue represents the head group $N^{+}(CH_3)_3$ and green represents the tail molecules CH_3/CH_2.

Surface atoms (Si and O) are represented by yellow (Si) and red (O). The water molecules are not shown in these figures for clarity.

Figs. 4-19 to 4-32 show stepwise evolution of monolayer adsorption on hydrophilic negatively charged silica surface.

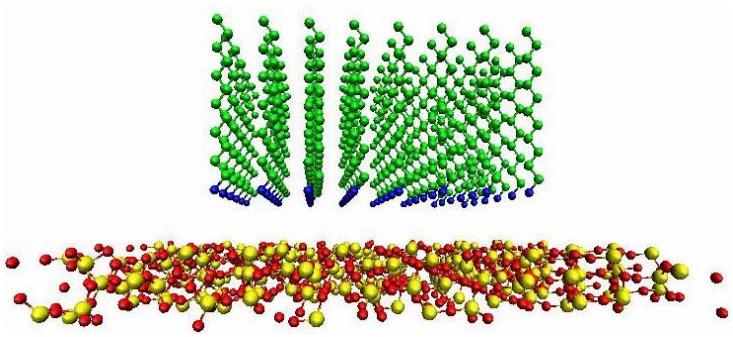


Figure 4-19: Initial structure at 0ps-Monolayer of 48 surfactants is placed above silica surface. The head groups of all the surfactants are facing towards the surface.

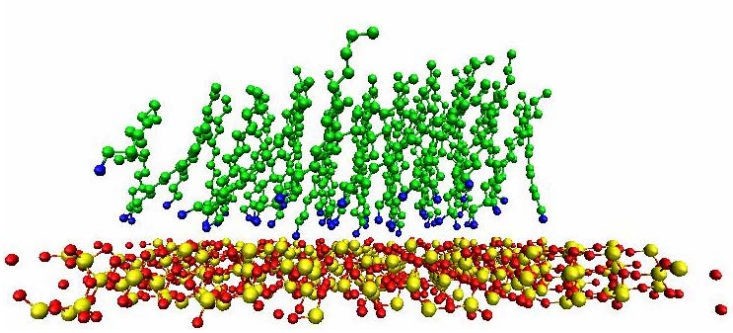


Figure 4-20: Structure at 60ps-All the cationic head groups of surfactant adsorb on negatively charged silica surface. All the tails bundle up together due to hydrophobic interaction.

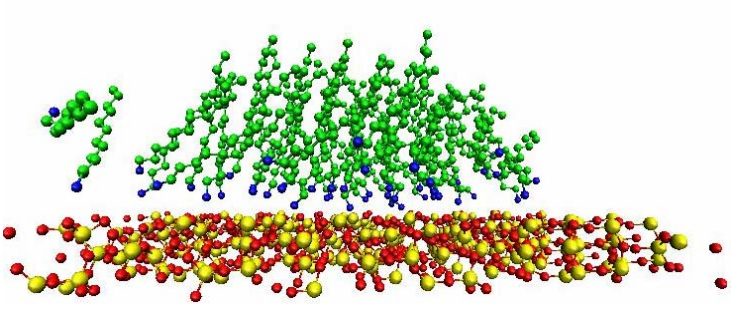


Figure 4-21: Structure at 80ps-Structure starts spreading on the surface. Chains of surfactant start bunching together to shield themselves from water molecules (hydrophobic interaction).

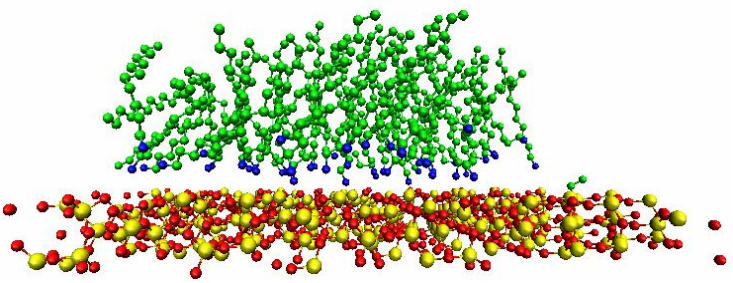


Figure 4-22: Structure at 100ps-Chains of surfactant try to reorient such that they can shield water molecules efficiently.

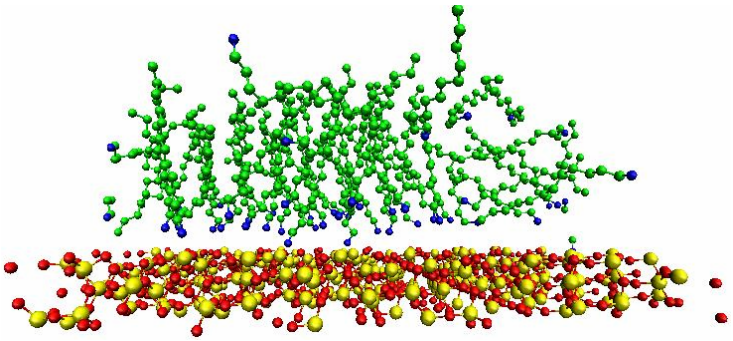


Figure 4-23: Structure at 180ps-Head groups start wrapping around the structure so that chains can be shielded. Also, head groups like to stay close to water molecules (hydrophilic interaction).

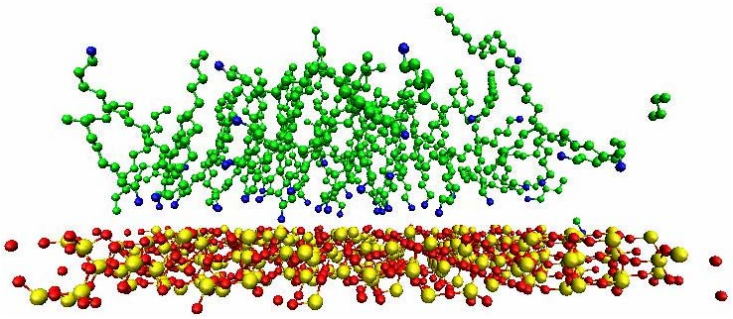


Figure 4-24: Structure at 190ps-More head groups can be seen on the other side of the surface to form elliptical micelle structure adsorbed on the silica.

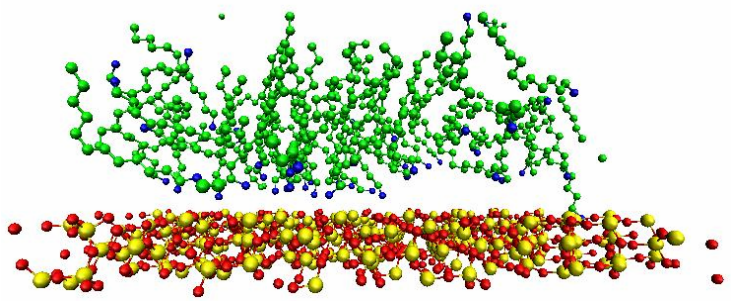


Figure 4-25: Structure at 215ps-Clear curvature can be seen in the structure with elliptical shape of micelle adsorbed on the silica.

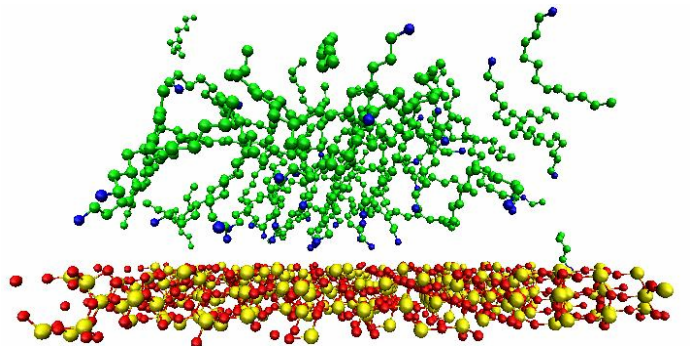


Figure 4-26: Structure at 250ps-Flat elliptical micelle structure stays adsorbed on the silica.

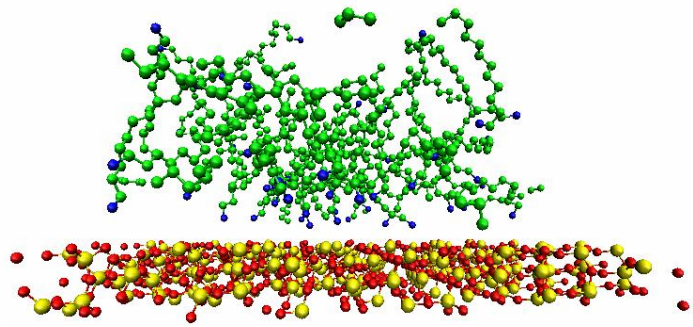


Figure 4-27: Structure at 265ps-Flat elliptical micelle structure stays adsorbed on the silica.

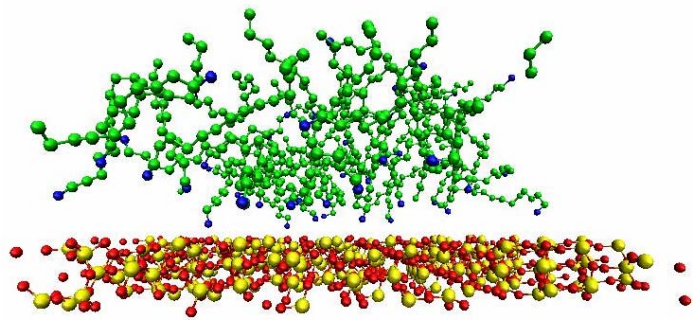


Figure 4-28: Structure at 300ps-Flat elliptical micelle structure stays adsorbed on the silica. No surfactants leave the structure.

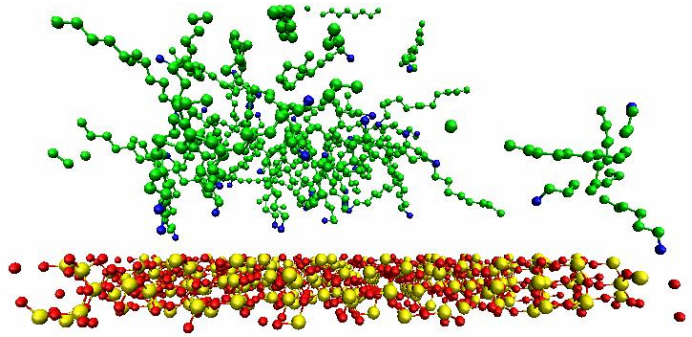


Figure 4-29: Structure at 395ps-Some surfactants detach from the structure having head groups attached on the surface.

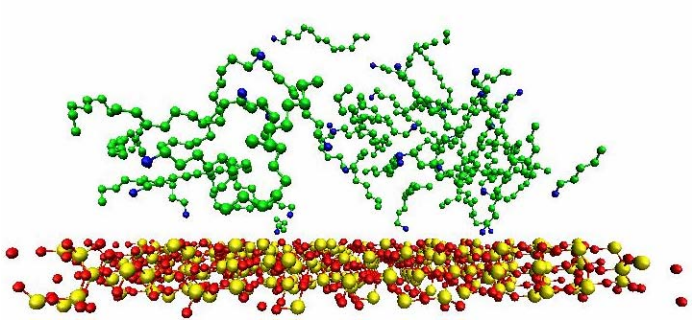


Figure 4-30: Structure at 675ps-Flat elliptical micelle stays adsorbed; all the surfactants are part of the structure.

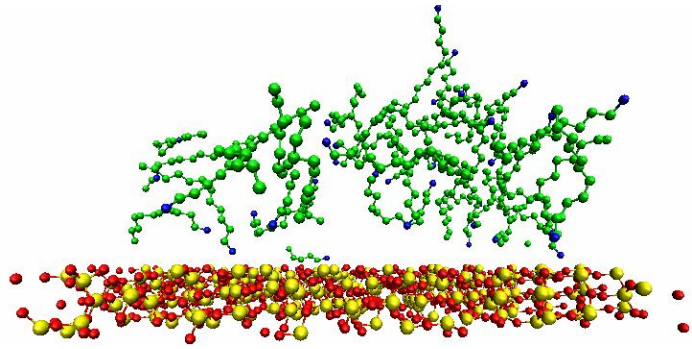


Figure 4-31: Structure at 700ps-Flat elliptical micelle stays adsorbed on silica surface.

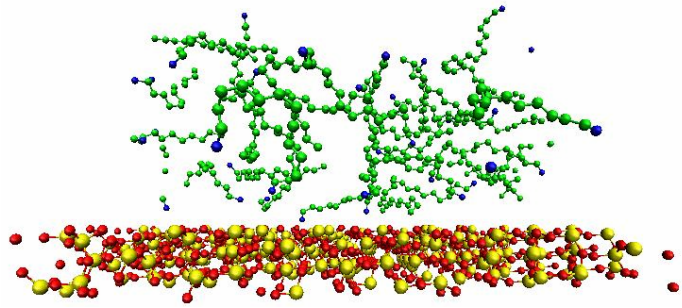


Figure 4-32: Structure at 725ps-Flat elliptical micelle stays adsorbed on the silica surface.

4.1.3 Bilayer on Silica

The simulation, shown in Fig. 4-33 to 4-38, was carried out with the bilayer of 48-surfactant aggregate structure placed on hydrophilic negatively charged silica surface. The bilayer was placed 6 Å from the negatively charged silica surface with dimensions of 60 Å on each side within the plane of the surface and a slab thickness of 5 Å. The temperature of the system is maintained at 300 K by application of the velocity rescaling method to all the atoms in the system except the surface atoms, which are held rigid.

The simulation predicts that the bilayer adsorbs on the silica and evolves into a flat elliptical micelle without any connections to adjacent micelles through the applied periodic boundary conditions, which is consistent with experimental data. In the experimental results with AFM shown in Fig. 4-1, small dots are micelles adsorbed on silica and there is not connection between these dots. While in simulations, the adsorbed micelle does no extend at the side, which can be interpreted by no connection between adjacent micelles through applied periodic boundary conditions. The head groups are columbically attracted to the oppositely charged sites on the silica surface.

In each figure, dark blue represents the head group $N^{+}(CH_3)_3$ and green represents the tail molecules CH_3/CH_2. Surface atoms (Si and O) are represented by yellow (Si) and red (O). The water molecules are not shown in these figures for clarity.

Figs. 4-33 to 4-38 show stepwise evolution of bilayer adsorption on hydrophilic negatively charged silica surface. Evolution of the bilayer structure on hydrophilic negatively charged silica surface is being tracked.

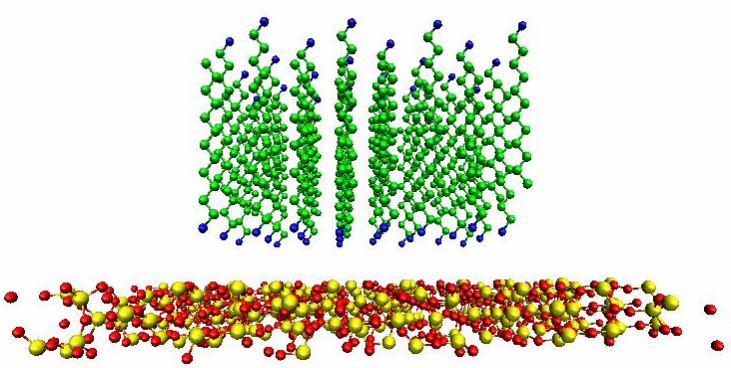


Figure 4-33: Initial structure at 0ps-Bilayer of 48 surfactants is placed on silica surface. Half of the head groups face the surface and rest of the head groups are in opposite direction.

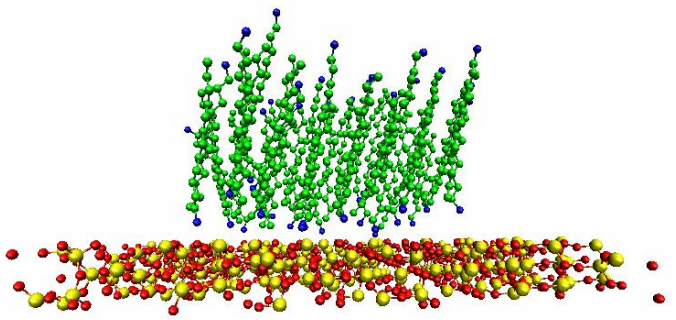


Figure 4-34: Structure at 15ps-Cationic head groups start adsorbing to the negatively charged silica surface.

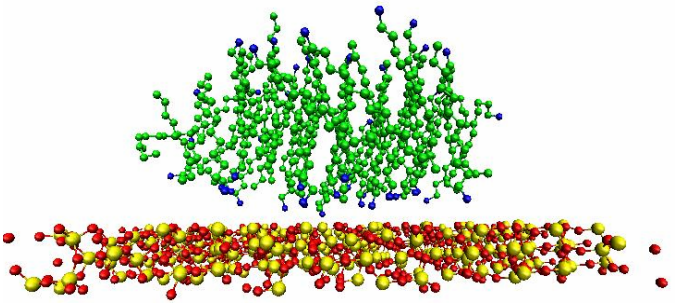


Figure 4-35: Structure at 90ps-As half of them are already oriented towards aqueous media, it takes less time for the structure as a whole to reach the lowest energy shape. An elliptical structure has been formed in which tails shield themselves inside while the head groups are oriented towards the water.

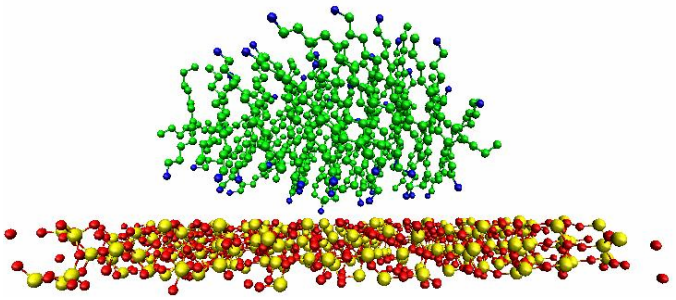


Figure 4-36: Structure at 100ps-Compact elliptical micelle structure stays adsorbed on silica.

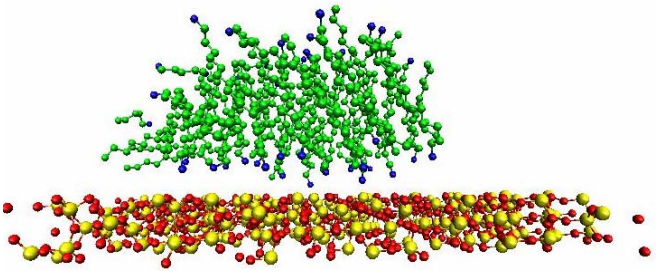
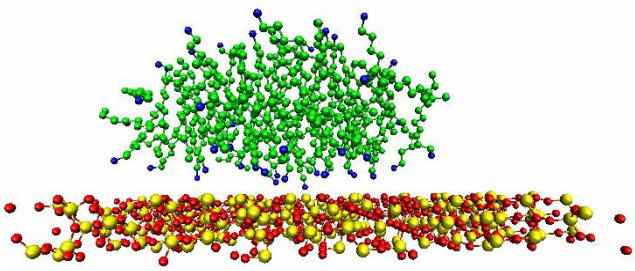
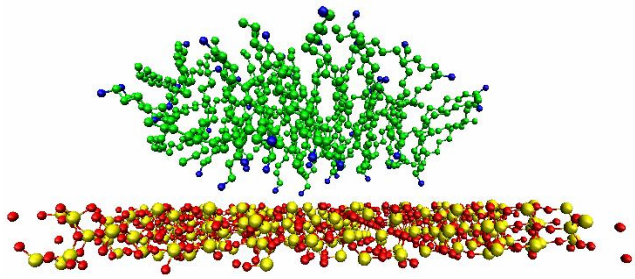


Figure 4-37: Structure at 120ps-Structure starts spreading on the silica surface forming a flat elliptical shape.

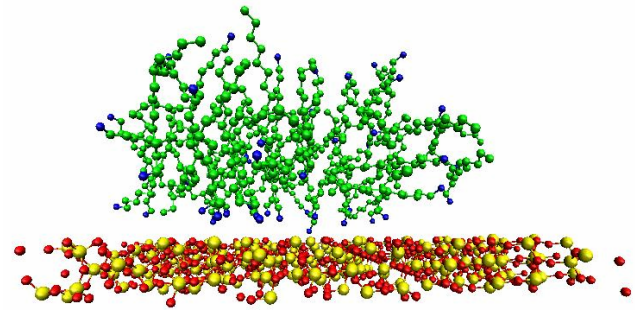
(a)



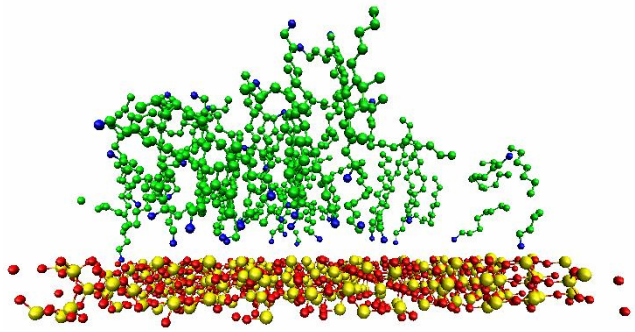
(b)



(c)

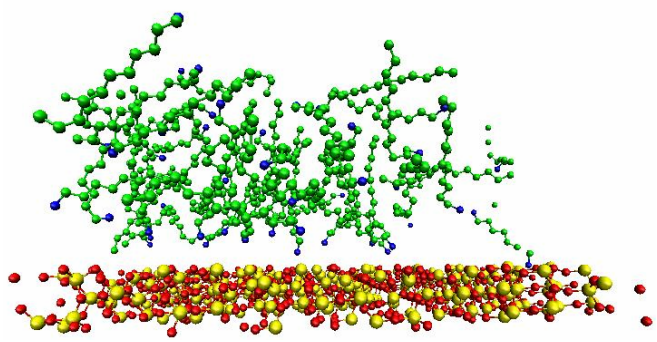


(d)

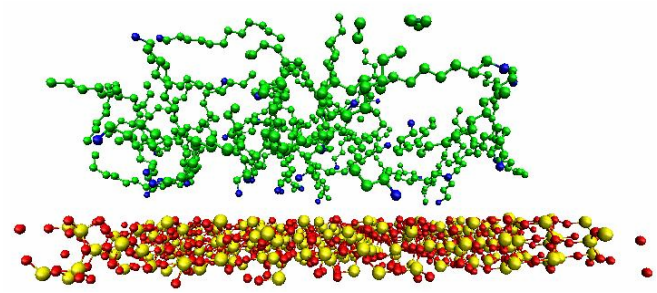


(e)

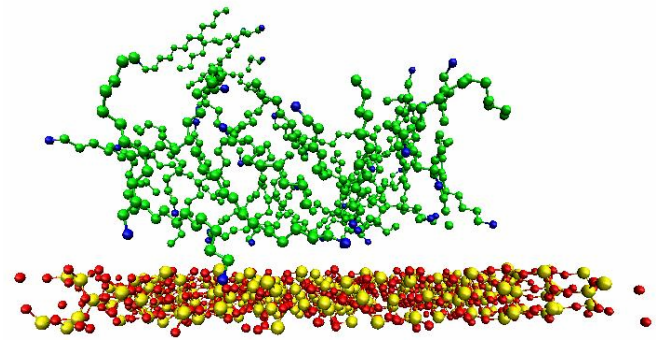
Figure 4-38: a-i shows evolution of the structure from 160ps to 580ps-The compact elliptical micelle structure stays adsorbed on silica. (a)- Structure at 160ps (b)- Structure at 190ps (c)- Structure at 215ps (d)- Structure at 255ps (e)- Structure at 285ps (f)- Structure at 340ps (g)- Structure at 490ps (h)- Structure at 550ps (i)- Structure at 580ps



(f)



(g)



(h)

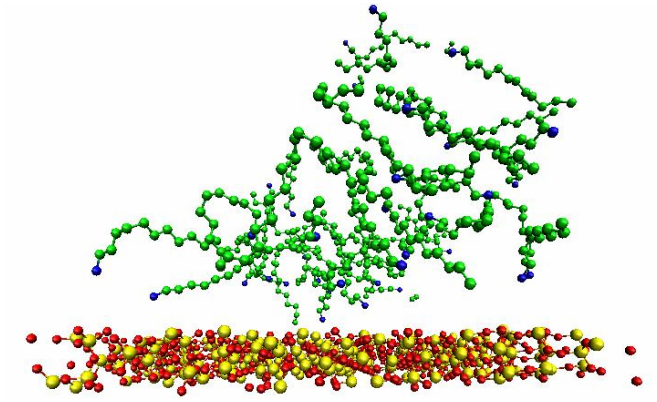


Figure 4-38 Continued

(i)

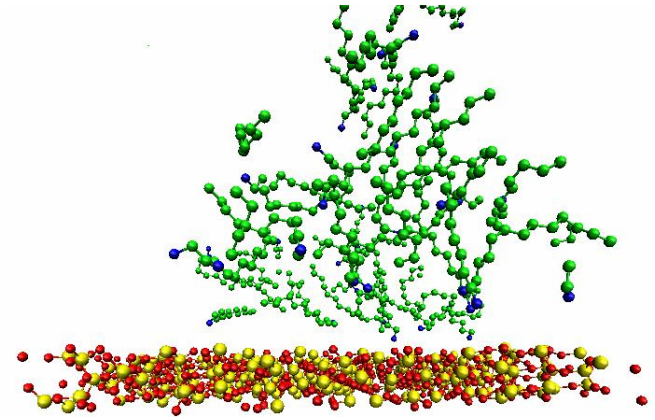
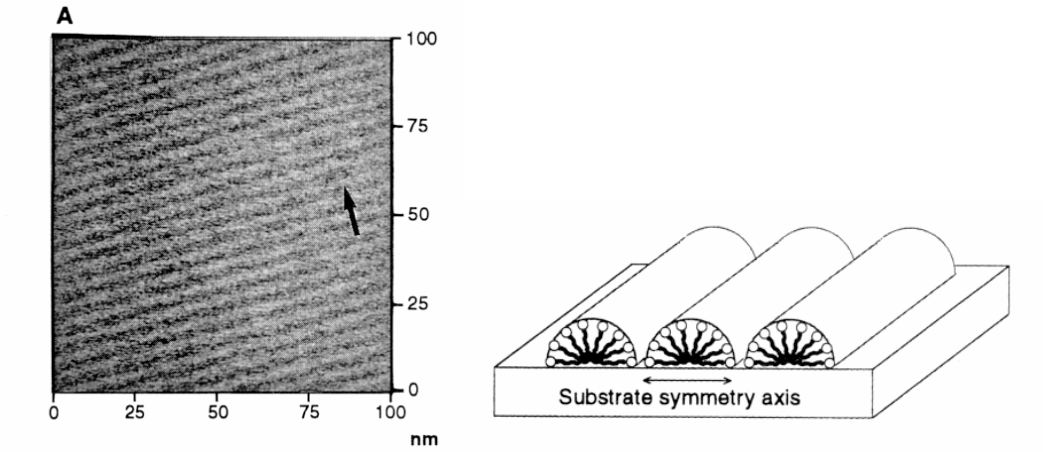


Figure 4-38 Continued

4.2 At the Graphite/Liquid Interface

The following simulation was designed to study the discrete self-aggregation of surfactants formed on graphite surfaces and to better understand experimental data from AFM[1-3]. Results AFM technique indicates that C_{12}TAB surfactants form hemi-cylindrical micelles on the hydrophobic graphite surface (shown in Fig. 4-39 a & b) with chains of surfactants lied down on surface due to strong hydrophobic attraction.



(a) (b)

Figure 4-39 C14TAB Surfactants aggregation on graphite surface (a): AFM images of aggregates of cationic surfactants on graphite (concentration =7 mM)[1] (b) Proposed model of hemi-cylindrical micelle on graphite surface[1]

The simulation, shown in Fig. 4-40 to 4-46, was carried out with a monolayer of 48-surfactants placed on graphite surface. The monolayer was placed 6 Å from the hydrophobic graphite surface (single sheet) with dimensions of 60 Å on each side within the plane of the surface. The temperature of the system is maintained at 300 K by application of the velocity rescaling method to all the atoms in the system except the surface atoms, which are held rigid.

The simulation predicts that the surfactants in the monolayer start adsorbing with chains of surfactants lying down on the surface due to their strong hydrophobic attraction, leaving head group standing up towards the water molecules. Finally, the adsorbed structure takes the shape of hemi-cylindrical micelle. In each figure, dark blue represents the head group $N^{+}(CH_3)_3$ and green represents the tail molecules CH_3/CH_2 and C of graphite surface.

Figs. 4-40 to 4-46 show stepwise evolution of monolayer adsorption on hydrophobic graphite surface. Evolution of the monolayer structure on graphite is being tracked.

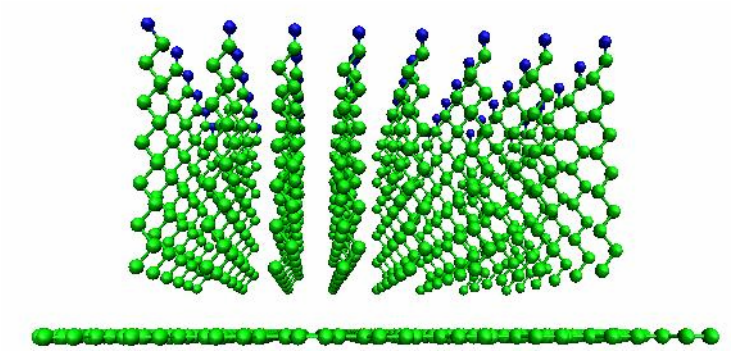


Figure 4-40: Initial structure at 0ps-Monolayer is placed on graphite surface. All the head groups are away from the surface and towards water molecules.All the tails are oriented towards the surface.

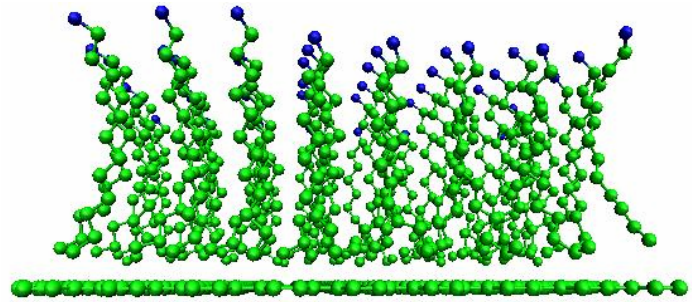


Figure 4-41: Structure at 3ps-Tails start adsorbing on graphite surface due to hydrophobic attraction.

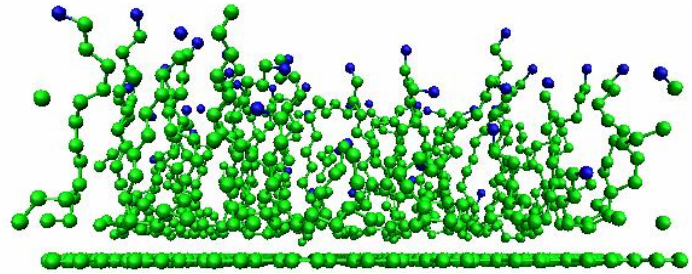


Figure 4-42: Structure at 5ps-Height of the structure is reducing as all the CH2/CH3 molecules have strong affinity with graphite surface.

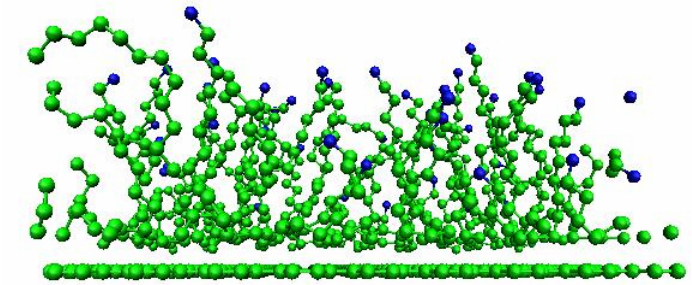


Figure 4-43: Structure at 8ps-All the tails lie down on the graphite surface keeping head group standing up towards water molecules, this results in a hemi-cylindrical structure which is in agreement with experimental data[1].

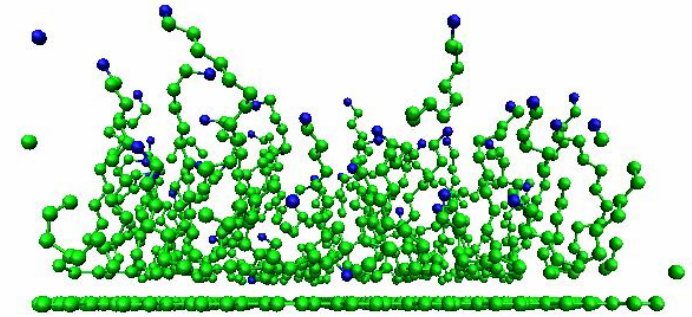


Figure 4-44: Structure at 20ps-Hemi-cylindrical structure stays adsorbed on graphite surface.

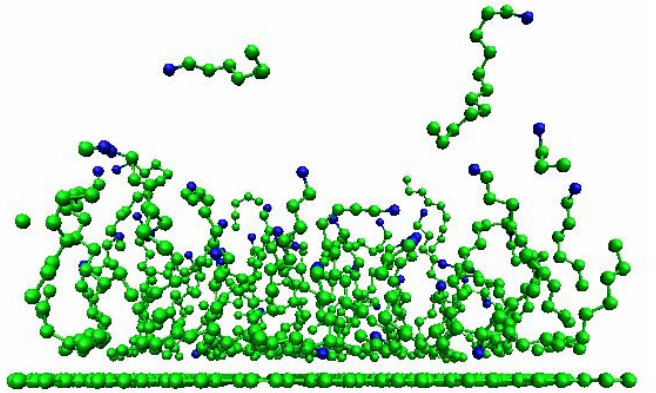


Figure 4-45: Structure at 25ps-Several surfactants appear to leave the structure. Hemi cylindrical structure stays adsorbed on graphite surface.

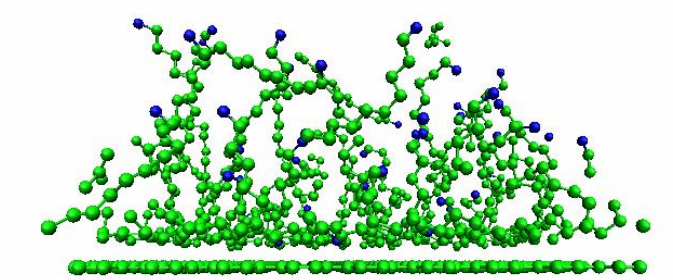


Figure 4-46: Structure at 31ps-Hemi cylindrical structure stays adsorbed on graphite surface. All the surfactants are part of the hemi cylindrical micelle structure on graphite surface.

4.3 Summary

As discussed in section 1.1, surfactant structures at the solid/liquid interface are increasingly being utilized as dispersants at high electrolyte concentrations, where conventional dispersants, e.g. inorganic dispersants and polymers, may not perform well. For example, in the chemical mechanical polishing of silicon wafers, nanometer-scale silica particles are used as polishing media. The presence of even small aggregates of these particles can create scratches on the surface and damage the silicon wafer. Thus, it is important to maintain a uniform dispersion of silica particles. In the case of nanoparticles, maintenance of dispersion becomes challenging because of inadequate knowledge and understanding of dispersion and adsorption. Hence, the development of better performing dispersants requires a fundamental understanding of the adsorption

mechanism of dispersants on the solid at the molecular level. As there are few limitations to the experimental techniques, it is hard to achieve atomic/molecular level understanding of adsorption of surfactants at solid-liquid interface. Computer simulations can be very helpful tool to achieve so. In this study we are using MD simulation to study kinetics, equilibrium structure and adsorption of micellar structure at silica and graphite surfaces.

The simulations predict that the round micelle adsorbs onto the silica surface without any connections to adjacent micelles through the applied periodic boundary conditions, which is consistent with experimental data. As the simulation evolves, the structure flattens into an elliptical shape, as shown in Fig. 4-2, and the head groups are columbically attracted to the oppositely charged sites on the silica surface. In the simulation of monolayer at silica, the results predict that the monolayer adsorbs on the silica, head groups start wrapping around the structure away from the surface and evolves in to flat elliptical micelle without any connections to adjacent micelles through the applied periodic boundary conditions. The head groups are columbically attracted to the oppositely charged sites on the silica surface. In the simulation of bilayer at silica, the results predict that the bilayer adsorbs on the silica and evolves into a flat elliptical micelle without any connections to adjacent micelles through the applied periodic boundary conditions, which is consistent with experimental data. The head groups are columbically attracted to the oppositely charged sites on the silica surface.

In the simulation of monolayer on graphite, the results predict that the surfactants in the monolayer start adsorbing with chains of surfactants lying down on the surface due to their strong hydrophobic attraction, leaving head group standing up towards the water molecules. Finally, the adsorbed structure takes the shape of hemi-cylindrical micelle.

Molecular level computer simulations such as those reported here complement experimental methods by providing molecular level insight into the adsorption of surfactants aggregate at solid-liquid interface.

CHAPTER 5
INDENTATION OF C_{12}TAB MICELLE

5.1 Mechanical Properties of Micelles at the Silica/Liquid Interface

In the previous chapter, we have discussed the self-aggregated structure of surfactants at liquid-solid interfaces, which is applicable to the dispersion of nanoparticles. It was shown that the strength and the rigidity of this surface aggregates is critical for creating steric repulsive forces which lead to good stabilization and dispersion of nanoparticles.[2-4] Also, mechanical properties of these surface aggregates can be varied by manipulating intermolecular forces. Additionally, the surface plays a catalytic role in the self-assembly of surfactants. Depending on the nature of the substrate (hydrophobic or hydrophilic), these surface aggregates can have different mechanical properties.

Subramanian-Ducker[45] has experimentally investigated the specific energy of C_{12}TAB aggregates on silica by AFM with force-distance curve using a glass particle as the probe. Moudgil et al. [2-4] conducted similar work in a more precise and reliable fashion to obtain mechanical properties and energy of the surface aggregates by experimentally measured force-distance curve of C_{12}TAB aggregates on silica using AFM. The experimental force-distance graphs (F vs. H) for interaction between AFM tip and surface aggregates on silica are shown in Fig. 5-1.

Several AFM experiments have been carried out[2-4] to study the mechanical properties of adsorbed micelles at liquid/silica interfaces. These studies conclude that in the presence of surfactants, such as 32 mM C_{12}TAB, the tip feels repulsion at a separation from the surface of 7 nm. This repulsion increases as the tip approaches the surface,

which can be interpreted as the compression of the adsorbed micelle structure on the surface. At around 3 nm of separation, the tip experiences a sudden attraction to the surface.

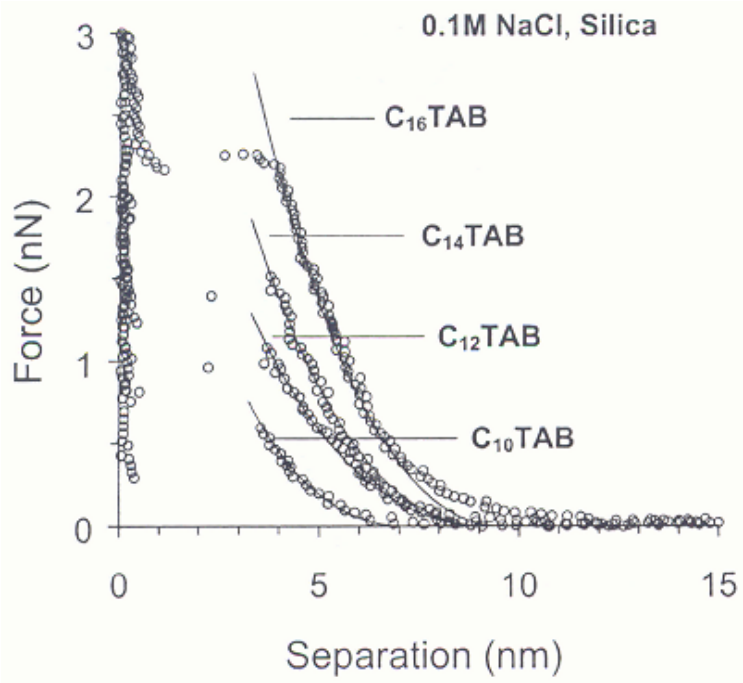


Figure 5-1: Experimental force-distance curve (circles) measured between AFM tip and CnTAB aggregates on silica substrate at pH= 5.8 in 0.1 M NaCl solution of CnTAB at 168; 32; 7.2 and 1.8mM for n=10, 12, 14, and 16 respectively. The concentrations are equal twice the cmc for CnTAB in DI respectively.[2-4]

Two interpretations of this data can be made. The first is that the tip breaks the structure (the mechanical strength of the micelle is measured to be about 1.5 nN). The second is that the adsorbed micelle structure just slips away from the location between tip and the surface, which allows the tip to be attracted to the surface. The experimental data does not provide any information on what is occurring at tip-surface distances of 2.6 nm-0.5 nm.

To shed more light on these interpretations, an MD simulation is used to indent the adsorbed micelle structure on silica with another silica surface as an indenter, as shown in Fig. 5-2.

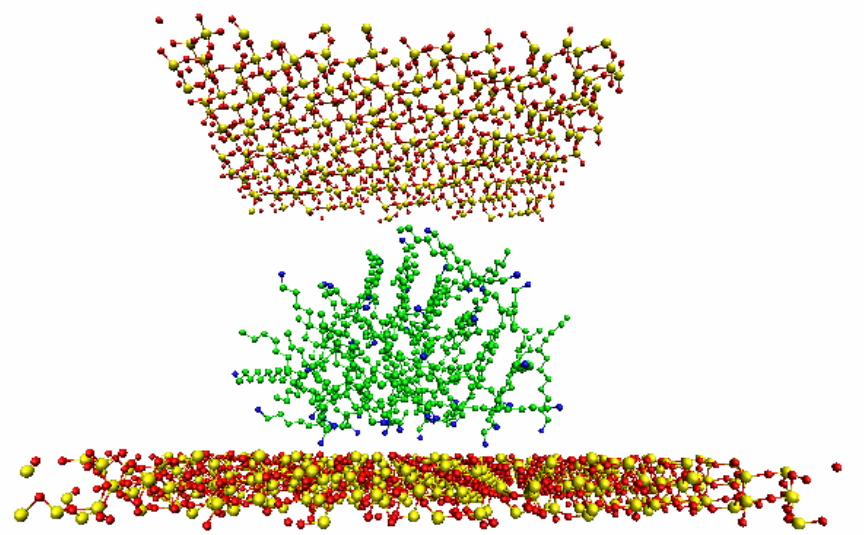


Figure 5-2: Indentation of adsorbed micelle on silica by another silica surface.

The indenter of 5 nm (top silica surface) is lowered at a constant velocity of 25 m/sec (0.00025 Å/fs) and compresses the micelle structure. The force felt by the indenter is calculated with respect to the distance between the indenter (top silica surface) and the substrate (bottom silica surface). The temperature of the system is maintained at 300 K by application of the velocity rescaling method to all the atoms in the system.Figs. 5-3 to 5-12 show the stepwise indentation process predicted in the simulations as the indenter is lowered at the speed of 25 m/sec towards the adsorbed micelle structure on silica.

Micelle Indentation on Silica (25 m/sec)

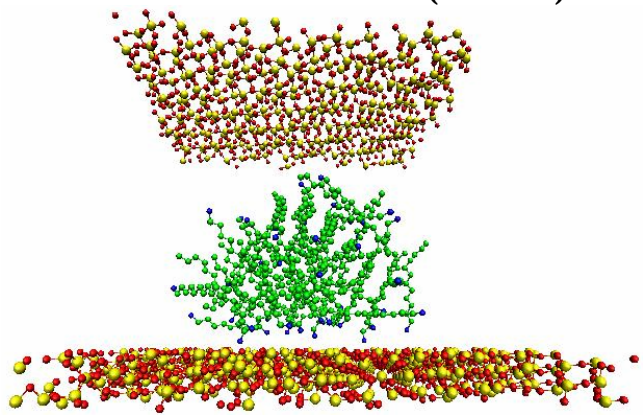


Figure 5-3: Distance between indenter and substrate = 35 Å-Initial structure as shown in Fig. 5-2

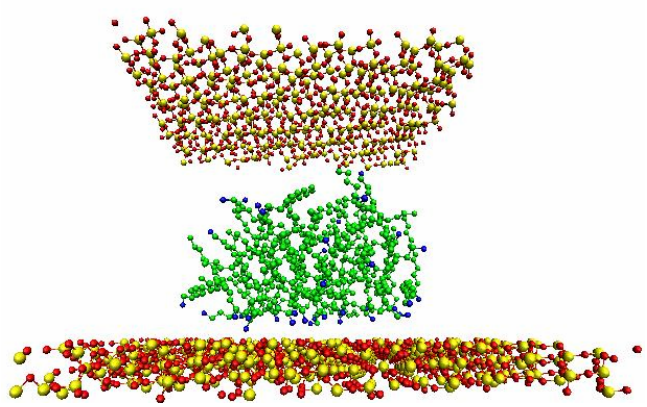


Figure 5-4: Distance between indenter and substrate = 32.8 Å-Shape of the micelle is still intact. No activity can be seen.

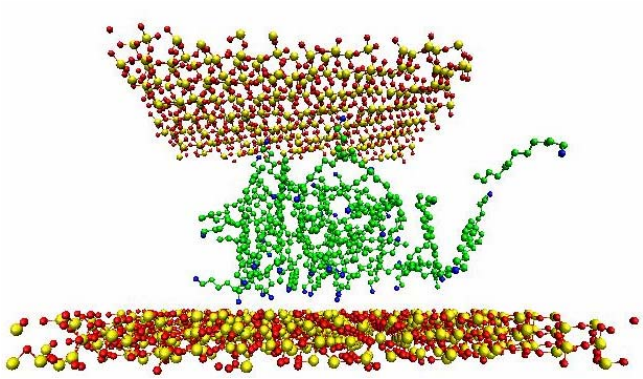


Figure 5-5: Distance between indenter and substrate = 28 Å-Cationic head groups of the surfactants near the indenter start adsorbing on the negatively charged silica indenter. The micelle starts feeling the presence of the indenter as the structure begins to compress; some of the chains start coming off the structure.

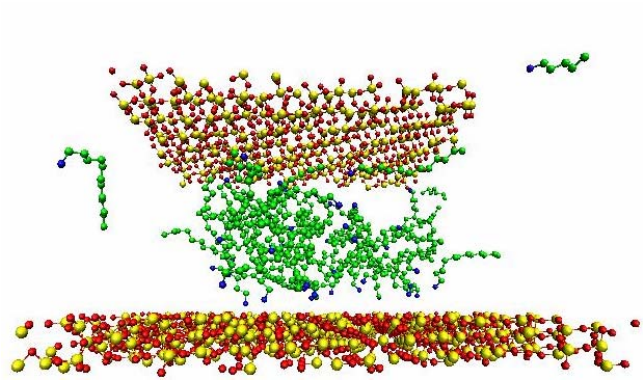


Figure 5-6: Distance between indenter and substrate = 23.7 Å-The structure is seen to be compressed more with multiple surfactants leaving the micelle. The micelle structure is squeezed between the indenter and the substrate.

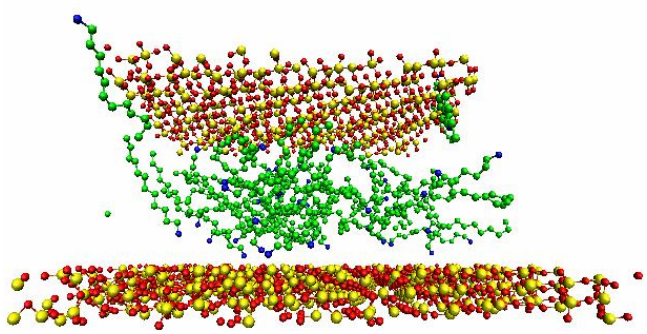


Figure 5-7: Distance between indenter and substrate = 20.3 Å-The micelle is still being compressed. As the width of the indenter is smaller than the substrate some chains leave the area near the surface around the sides of the indenter.

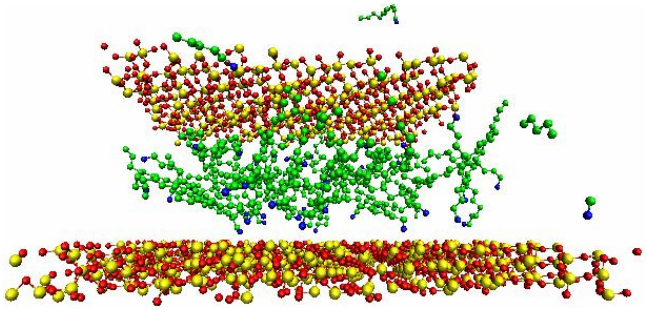


Figure 5-8: Distance between indenter and substrate = 17.7 Å-More chains are breaking away from the region between the indenter and the substrate as the distance is decreased between them; some chains can be seen above the indenter.

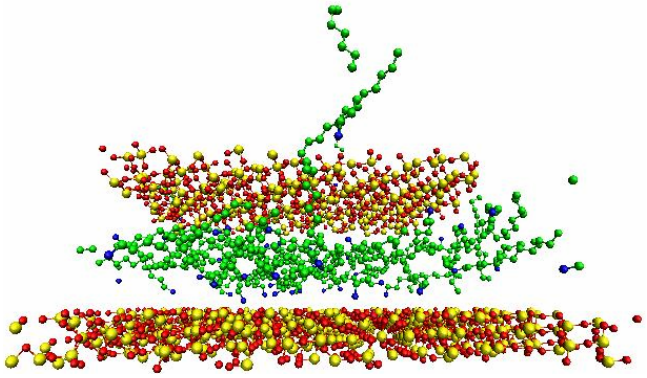


Figure 5-9: Distance between indenter and substrate = 13.7 Å-Surfactants continue to leave the region between the indenter and the substrate.

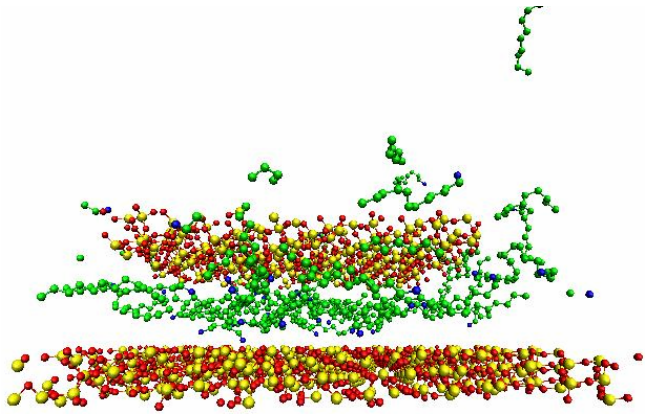


Figure 5-10: Distance between indenter and substrate = 10.8 Å-Surfactants continue to leave the region between the indenter and the substrate.

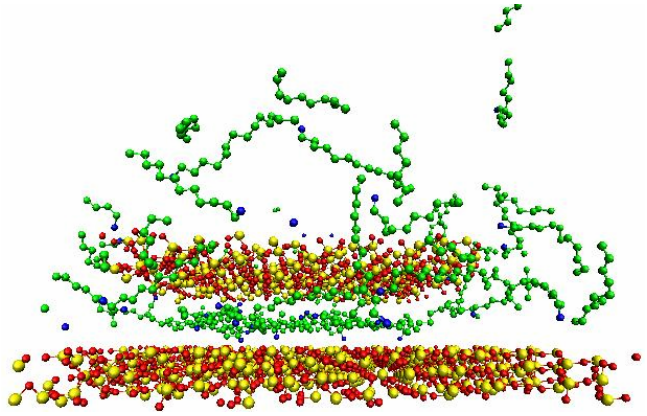


Figure 5-11: Distance between indenter and substrate = 8 Å-Surfactants continue to leave the region between the indenter and the substrate.

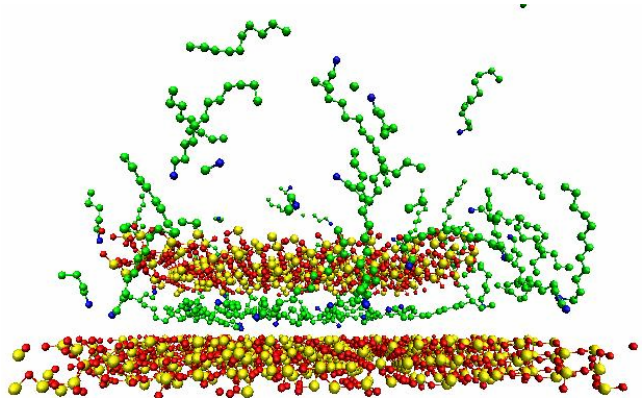


Figure 5-12: Distance between indenter and substrate = 7.45 Å-There are still some surfactants trapped between the indenter and the substrate; the rest of the surfactants have already left the region between indenter and substrate.

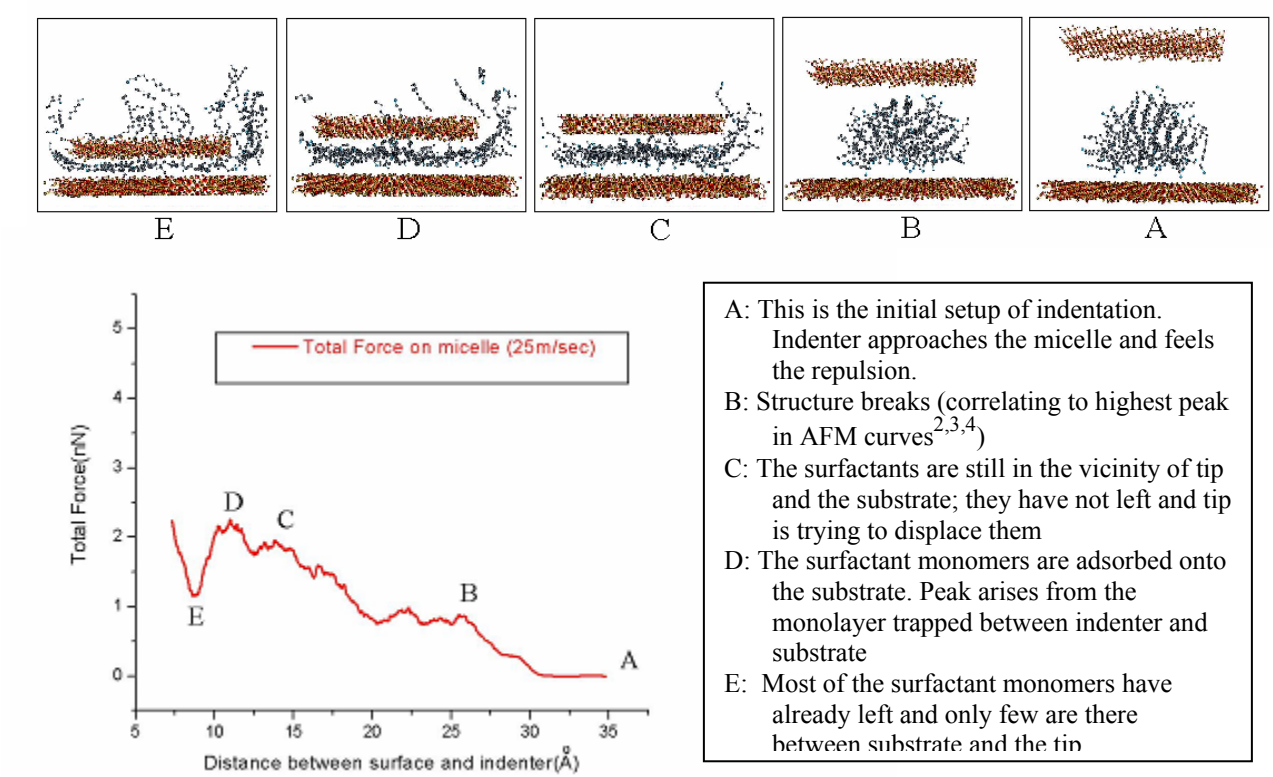


Figure 5-13: Summary of the results of MD simulations of indentation of the micelle-covered silica surface with graph of total force vs. distance between surface and indenter.

The simulations predict that the micelle structure breaks apart when the distance between the indenter and the surface is 26 Å, as shown in Fig. 5-13. This result is in good agreement with the experimental findings discussed above. The simulations also conclusively show that the tip indentation process breaks the micelle structure (see peaks B, C, D). Experimental data shows that the force required to break the structure is 1.5 nN[12] while MD simulations predict the force required to break apart the micelle is 1 nN (see Fig. 5-13). These results are in excellent qualitative agreement. The difference in the quantitative values can be explained by the smaller AFM tip (diameter of about 5 nm) used in simulations relative to the much larger experimental tips.

The MD simulation of the indentation process presented in Fig. 5-13 also shows the presence of small peaks in the force curve after the micelle structure breaks apart. The

presence of peak C is explained by the fact that once the structure is broken the surfactant monomers are still between the tip and the surface and are not able to escape. Consequently, after the micelle breaks apart the force increases. This trapping of surfactants to some extent is due to the relatively rapid indentation rate of 25 m/s which allows much less time for the surfactants to escape than in the experiments, where the tip velocity is on the order of μm/s. The presence of peak D can be explained by monomers adsorbed on the substrate and the force required to get rid of this last monomer layer from the substrate.

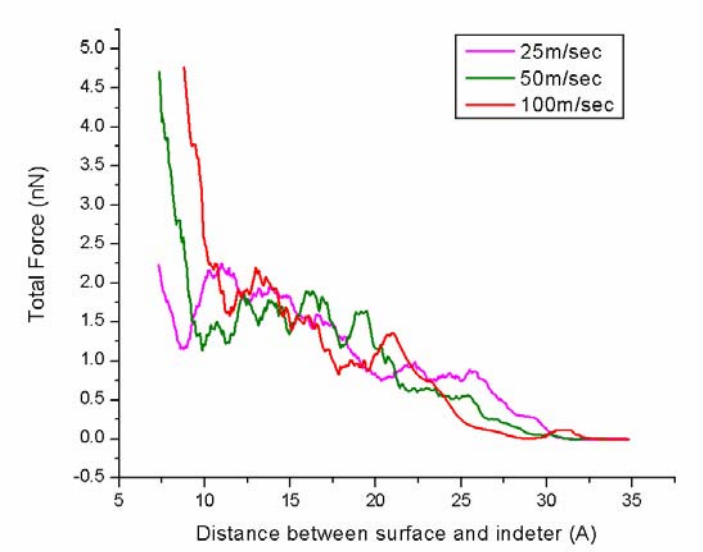


Figure 5-14: Effect of different indentation rates on the predicted force-distance curve.

Fig. 5-14 shows that over the rates that are accessible in these classical MD simulations, the effect of the rate of indentation on the force to break the adsorbed micelle is negligible. Each curve shows the same trends. The same events discussed above occur in all the simulations regardless of the rate of indentation. However, at the highest indentation rate of 100 m/sec, the first peak where the micelle breaks occurs at 21 Å whereas the other rates predict breakage at a tip-surface distance of 26 Å (see Fig. 5-14). This difference can be explained by the higher compression results achieved during

the higher indentation rates and illustrate the effect of increased number of surfactants trapped between the indenter and the surface at high indentation rates.

5.2 Mechanical Properties of Micelles at the Graphite/Liquid Interface

No experimental work has been yet been done to study the mechanical properties and energy of surfactant aggregates on graphite surfaces. To shed more light on mechanical properties of micelles on graphite, an MD simulation is used to indent the adsorbed micelle structure on graphite with another graphite surface as an indenter, as shown in Fig. 5-15.

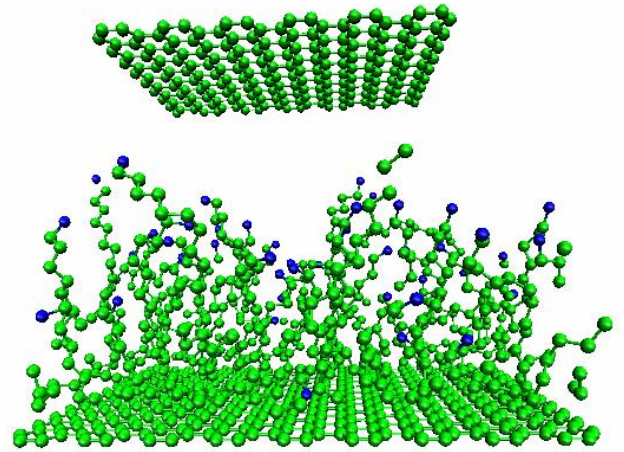


Figure 5-15: Indentation of adsorbed micelle on graphite by another graphite surface.

The indenter (top graphite surface) is lowered at a constant velocity of 50 m/sec (0.0005 Å/fs) and compresses the micelle structure. The force felt by the indenter is calculated with respect to the distance between the indenter (top graphite surface) and substrate (bottom graphite surface). The temperature of the system is maintained at 300 K by application of the velocity rescaling method to all the atoms in the system.

Figs. 5-16 to 5-22 show the stepwise indentation process as the indenter lowers at a rate of 50 m/sec towards the adsorbed micelle structure on graphite.

Micelle Indentation on Graphite (50 m/sec)

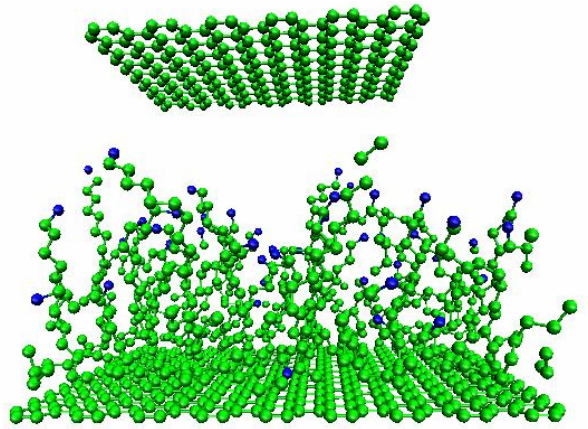


Figure 5-16: Distance between indenter and substrate = 27 Å-Initial structure shown in Fig. 15. Indenter (graphite) is being lowered towards the hemi-cylindrical micelle structure adsorbed on the graphite substrate.

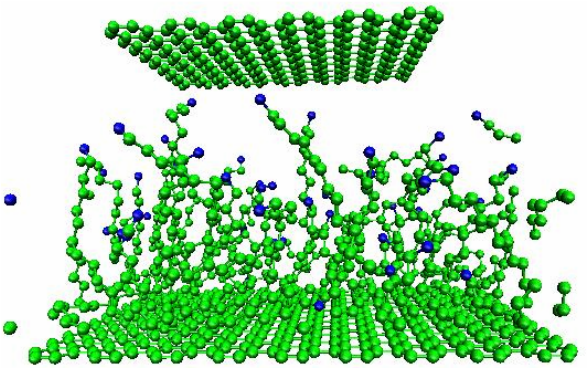


Figure 5-17: Distance between indenter and substrate = 23 Å-The adsorbed micelle structure starts feeling the presence of the graphite indenter as the structure begins to compress and break.Tails start adsorbing on to graphite indenter due to strong hydrophobic interaction.

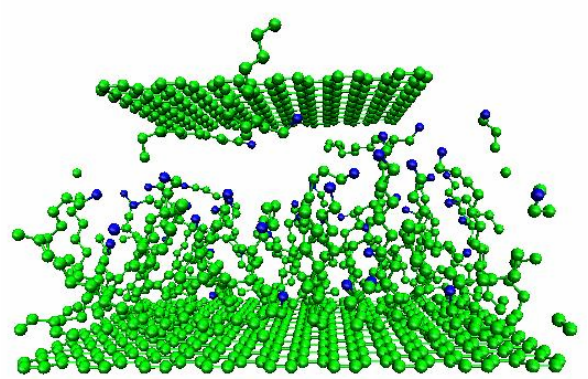


Figure 5-18: Distance between indenter and substrate = 19.5 Å-Adsorbed surfactants are getting squeezed further as the indenter approaches them. Some surfactants leave from the sides of the indenter, its width is smaller than the width of the substrate.

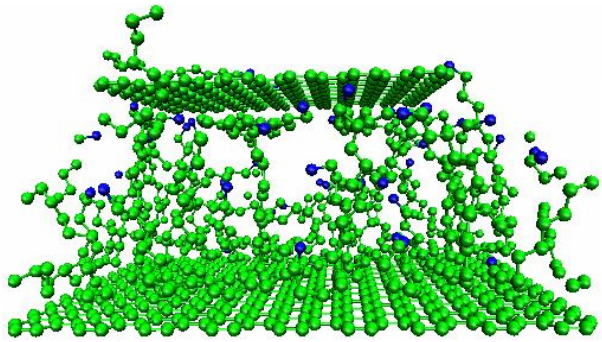


Figure 5-19: Distance between indenter and substrate = 15.75 Å-More surfactants adsorb on the indenter as well as leave from the region between the indenter and the substrate.

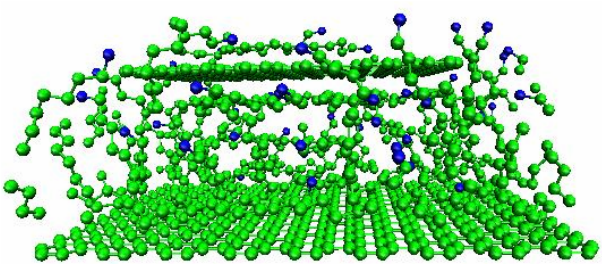


Figure 5-20: Distance between indenter and substrate = 12 Å-More surfactants leave from the region between the indenter and substrate; some of them are seen above the indenter, adsorbed hydrophobically.

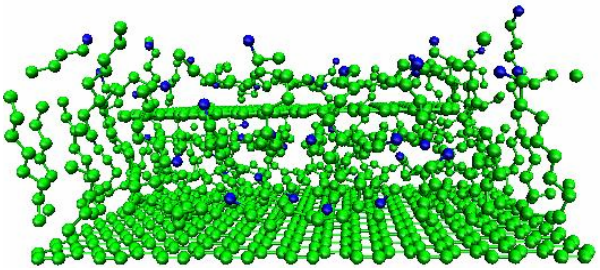


Figure 5-21: Distance between indenter and substrate = 9 Å-More surfactants leave the region between the indenter and substrate; some of them are seen above the indenter, adsorbed hydrophobically.

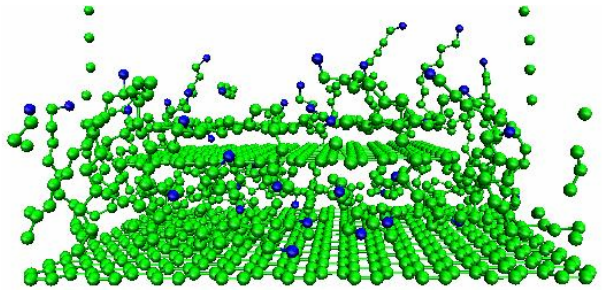


Figure 5-22: Distance between indenter and substrate = 7 Å-There are still some surfactants trapped between the indenter and the substrate when indentation is stopped at a tip-surface distance of 7 Å.

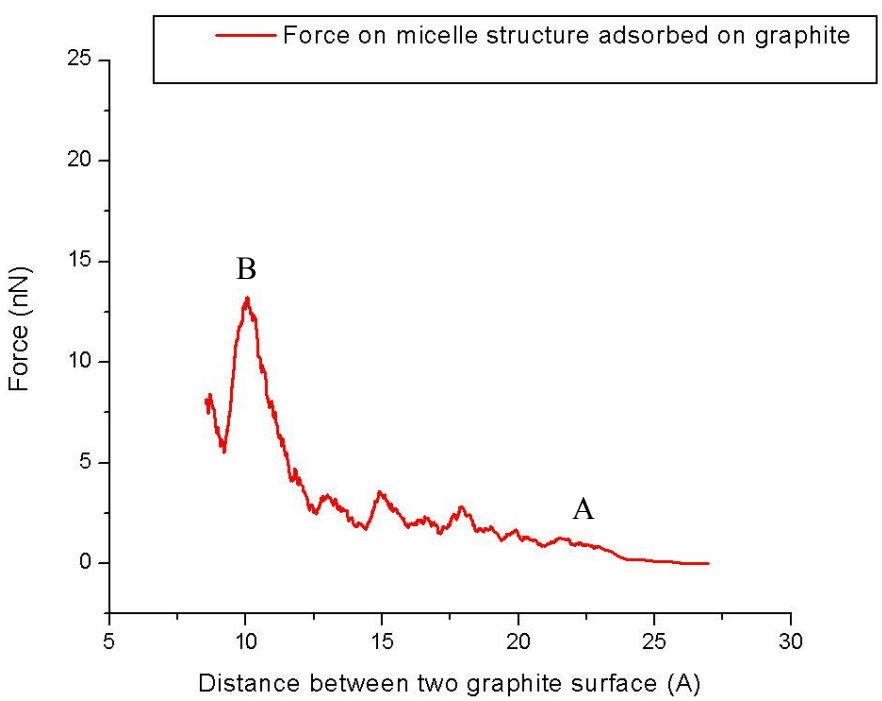


Figure 5-23: Total force vs. distance between surface and indenter graph obtained from results of MD simulations of indentation of the micelle-covered graphite surface.

The indentation results predict that the micelle structure breaks when the distance between the indenter and the graphite surface is 22.5 Å as shown as A in Fig. 5-23. This distance is smaller than in case of indentation of micelle on silica (26 Å). This difference can be explained by the difference in height between hemi-cylindrical structure of micelle on graphite and elliptical structure of micelle on silica. Indentation results indicate that the micelle structure breaks; the structure is not slipping away from the region between indenter and the substrate. The micelle structure breaks at an indentation force of 1.25 nN, which is slightly above the force predicted to break the micelle structure on silica (1 nN). This difference can be explained by a stronger interaction (hydrophobic) between the absorbed structure and the graphite substrate. Peak D (Fig.5-13) that is the highest repulsion felt by the indenter due to trapped surfactants is approximately 13 nN (Peak B, Fig. 5-23). This is much higher than the repulsion (2.25 nN) felt by silica indenter due to

surfactants trapped between indenter and silica substrate. This can be explained by the stronger interaction of surfactant tail adsorbed on graphite surface hydrophobically.

5.3 Summary

In many industrial processes, such as chemical mechanical polishing, mixing and flow of particulate suspensions, self-assembled structures of surfactants encounter appreciable loads and pressures. To design process conditions for optimal results, the mechanical properties of these aggregates must be controlled. The force required to disintegrate the micelles is the upper limiting applied load for dispersion. Above this load, the aggregate micelle structure is disintegrated and undesirable adhesion occurs between the two sliding surfaces. Over the last decade, several groups have used AFM to study the mechanical properties of micelles [3,4]. The advantage of using AFM is that the load required to break the micelles apart can be directly measured. The disadvantage of this approach is that the mechanisms involved in micelle disintegration cannot be resolved.

In this study we have used MD simulations to study the kinetics of the breakdown of adsorbed micelles during indentation with proximal probe tips. The simulations predict that the force required to break apart these adsorbed micelles through proximal probe indentation is about 1 nN, which correlates well with experimental values (1.5 nN). The tip-surface distance where the mechanically induced dissociation is predicted to occur is about 26 Å, which also correlates with experimental data (26 Å). Lastly, the simulations predict the presence of small peaks in the force curve that are not seen experimentally that is due to the presence of residual surfactants between the tip and the surface after the breakage of the micelle is achieved. The rates that are accessible in these classical MD

simulations, the effect of the rate of indentation on the force to break the adsorbed micelle is negligible.

In the case of indentation of the structure adsorbed on graphite surface by graphite indenter, the results predict that the micelle structure breaks when the distance between the indenter and the graphite surface is 22.5 Å. This distance is smaller than in case of indentation of micelle on silica (26 Å) due to smaller height of hemi-cylindrical structure on graphite than elliptical structure on silica. Indentation results indicate that the micelle structure breaks; the structure is not slipping away from the region between indenter and the substrate. The micelle structure breaks at an indentation force of 1.25 nN, which is slightly above the force predicted to break the micelle structure on silica (1 nN). This difference can be explained by a stronger interaction (hydrophobic) between the absorbed structure and the graphite substrate. The highest repulsion felt by the indenter due to trapped surfactants is approximately 13 nN. This is much higher than the repulsion (2.25 nN) felt by silica indenter due to surfactants trapped between indenter and silica substrate. This can be explained by the stronger interaction of surfactant tail adsorbed on graphite surface hydrophobically.

With the use of the MD simulations, the mechanical properties and breakage characteristics of adsorbed micelles on silica and graphite surfaces are known. This information will be helpful in deciding the operating condition in chemical mechanical polishing to keep the stable dispersed slurry throughout the process.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

6.1 General Conclusions

Emerging technologies such as controlled drug delivery, abrasives for precision polishing, coating, paints, and nano composite materials are increasingly relying on nano particulate material to achieve optimum performance. It is therefore very important to produce well dispersed nano particulate dispersions. As self aggregate surfactant structures are optimally used as dispersants, understanding the different structures that can occur in the bulk and on the dispersing particles is urgently needed. It is currently difficult to achieve atomic/molecular level understanding of evolution of micellar structure in the bulk using only experimental methods. Computer simulation is therefore a helpful tool in providing information that is complementary to experimental data in order to achieve this understanding at the atomic/molecular level. In this study we have used MD simulations to study the growth mechanism of micelles in water and the structural evolution of micelles in water and at water/solid surface interfaces with both silica and graphite surfaces. We have also studied the kinetics of the breakdown of adsorbed micelles during indentation with proximal probe tips.

The simulations indicate that self-aggregated surfactants form spherical or elliptical structures in water at concentrations much above the surfactant cmc. Within the structure, some layering of surfactants is also predicted to occur. This may be attributed to the strong hydrophobic interactions between the CH_3/CH_2 surfactant tails in the aqueous

media. The simulations with two clusters of 24 surfactants show initial exchanges of surfactants between the clusters; eventually two clusters coalesce and form a single micelle. Thus it follows the Smoluchowski model for micelle growth when the concentration of surfactant is much above the cmc.[19,20] In the simulation of surfactant monolayers in water the structure is far from equilibrium, where tails of surfactants are directly in contact with water molecules. As this system evolves towards the equilibrium structure, the surfactant tails rearrange within the structure and eventually form a spherical micelle with head groups on the surface of the sphere and tails coiled inside in a random manner. Similar results can be seen in case of bilayer surfactant structures in water.

In the chemical mechanical polishing of silicon wafers, nanometer-scale silica particles are used as the polishing media. The presence of even small aggregates of these particles can create scratches on the surface and damage the wafer. Thus, it is important to maintain a uniform dispersion of silica particles. In the case of nanoparticles, maintenance of dispersion becomes challenging because of inadequate knowledge and understanding of dispersion and adsorption. Hence, the development of better performing dispersants requires a fundamental understanding of the adsorption mechanism of dispersants on the solid at the molecular level. Here we are have used MD simulations to study the kinetics, micelle equilibrium structure, and structure of adsorbed micelles at the solid-liquid interface using both silica and graphite surfaces.

The simulations predict that spherical micelles adsorb onto silica surfaces without any connections to adjacent micelles even at high concentrations, which is consistent with experimental data. As the simulation evolves, the structure flattens into an elliptical

shape and the head groups are columbically attracted to the oppositely charged sites on the silica surface. In the simulation of surfactant monolayers on silica, the simulations predict that the monolayer adsorbs on the silica, head groups start wrapping around the structure away from the surface, and the micelle structure evolves into a flat elliptical micelle without any connections to adjacent micelles. Again, the head groups are columbically attracted to the oppositely charged sites on the silica surface. In the simulation of micelle bilayers on silica, similar results can be seen as the bilayer adsorbs and evolves into flat elliptical micelles on negatively charged silica surfaces without any connections to adjacent micelles, which is also consistent with experimental data. The head groups are columbically attracted to the oppositely charged sites on the silica surface.

In the simulation of surfactant monolayers on graphite, the simulations predict that the surfactants in the monolayer start adsorbing with chains of surfactants lying down on the surface due to their strong hydrophobic attraction, leaving head group oriented away from the surface towards the water molecules. Finally, the adsorbed structure takes the shape of hemi-cylindrical micelle.

In many industrial processes, such as chemical mechanical polishing, mixing and flow of particulate suspensions, self-assembled structures of surfactants encounter appreciable loads and pressures. To design process conditions for optimal results, the mechanical properties of these aggregates must be controlled. The force required to disintegrate the micelles is the upper limiting applied load for dispersion. Above this load, the aggregate micelle structure disintegrates and undesirable adhesion occurs between the two sliding surfaces. Over the last decade, several groups have used AFM to

study the mechanical properties of micelles.[3,4] The advantage of using AFM is that the load required to break the micelles apart can be directly measured. The disadvantage of this approach is that the mechanisms involved in micelle disintegration cannot be resolved.

In this study we have used MD simulations to study the kinetics of the breakdown of adsorbed micelles during indentation with proximal probe tips. The simulations predict that the force required to break apart these adsorbed micelles through proximal probe indentation is about 1 nN, which correlates well with experimental values (1.5 nN). The tip-surface distance where the mechanically induced dissociation is predicted to occur is about 26 Å, which also correlates with experimental data (26 Å). Lastly, the simulations predict the presence of small peaks in the force curve that are not seen experimentally that is due to the presence of residual surfactants between the tip and the surface after the breakage of the micelle is achieved. For the rates that are accessible in these classical MD simulations, the effect of the rate of indentation on the force to break the adsorbed micelle is negligible.

In the case of indentation of the structure adsorbed on graphite surface by graphite indenters, the results predict that the micelle structure breaks when the distance between the indenter and the graphite surface is 22.5 Å. This distance is smaller than in case of indentation of micelles on silica (26 Å) due to the smaller height of the hemi-cylindrical micelle structure on graphite relative to the elliptical micelle structure on silica. Indentation results indicate that the micelle structure breaks; the structure is not slipping away from the region between the indenter and the substrate. The micelle structure breaks at an indentation force of 1.25 nN, which is slightly above the force predicted to break the

micelle structure on silica (1 nN). This difference can be explained by a stronger interaction (hydrophobic) between the absorbed structure and the graphite substrate. The highest repulsion felt by the indenter due to trapped surfactants is approximately 13 nN. This is much higher than the repulsion (2.25 nN) felt by silica indenter due to surfactants trapped between the indenter and the silica substrate. This can be explained by the stronger interaction of surfactant tail adsorbed on the graphite surface hydrophobically.

With the use of the MD simulations, the mechanical properties and breakage characteristics of adsorbed micelles on silica and graphite surfaces are predicted. This information will be helpful in determining the optimum operating conditions required in chemical mechanical polishing to maintain a stable dispersed slurry throughout the process.

6.2 Future Work

In our study, every simulation runs for a few hundred picoseconds to few tens of nanoseconds. Although these times scales are too short to entirely understand the true dynamic properties of micelles and surfactants (changes and fluctuations in the shape of micelles and micelle aggregation number can vary from 10^{-11} to 10^{4} seconds), the simulations can provide information which is representative of the processes occurring at the dynamic level. MD simulation can be used to perform simulations only in the range of several nano seconds. It cannot be used to trace activities occurring in the range of micro and milli seconds. In order to lengthen the time scales, hybrid simulation methods such as those presented by Jacobsen et al.[83] would be used. In hybrid simulations, the MD simulation method is combined with the MC (Monte Carlo) simulation method. In this way dynamical information can be obtained in the nano second time scale with MD, while long-time-scale events such as changes and fluctuations in the shape of micelles,

micelle aggregation number, etc. can be obtained from the MC simulation. This hybrid simulation approach is computationally expensive but it allows for the study of aggregation and breakage of micelles in water and at solid-liquid interfaces in their entirety.

Another approach that is used to capture long-time-scale activity is the temperature-accelerator dynamics developed by Sørensen and Voter.[84] This method is a combination of conventional molecular dynamics and statistical dynamics and is based on harmonic transition state theory. In this method the thermally activated behavior of the system is studied by performing an MD simulation at higher temperature. Therefore, the rate of the activated process can be raised and the activities that would otherwise occur over longer time scales can be observed in the simulations. However, raising the temperature may induce some transitions that may not occur at ordinary temperatures. Such temperature-induced effects are corrected by allowing only transitions that should take place at ordinary temperatures. In this way the accessible time scale is extended by several orders of magnitude.

The indentation simulations predict the presence of residual surfactants between the tip (5 nm) and the surface after the breakage of the micelle is achieved. This trapping of surfactants to some extent is due to the relatively rapid indentation rate of 25 m/s which allows much less time for the surfactants to escape than in the experiments, where the tip velocity is on the order of μm/s. The figure shown below (Fig. 6-1) is the set up of the indentation process with a carbon nanotube (CNT) tip as an indenter. With CNT tips (that are about 1nm in diameter), surfactants will have more space to escape and it will be interesting to see if there are residual surfactants between CNT and the silica substrate.

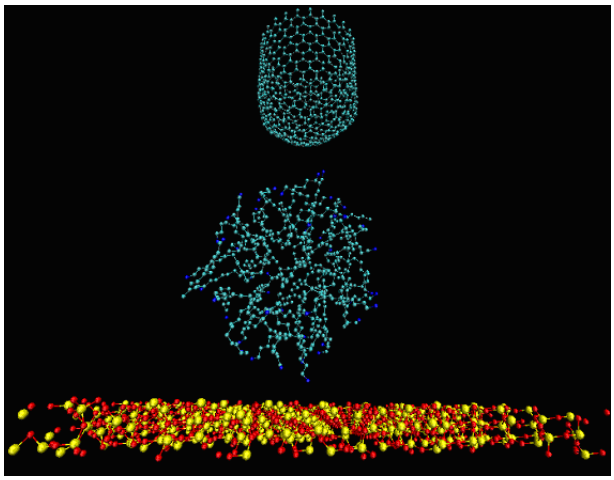


Figure 6-1: Indentation of adsorbed structure by CNT indenter

It is also very important to study the adhesion properties of adsorbed micelles on surfaces. The following setup (Fig. 6-2) allows for the determination of lateral forces by drawing the indenter of silica from the side. Lateral forces will be applied when the indentor approaches the adsorbed micellar structure on the surface. The lateral force felt by the silica indenter can be correlated to the adhesion properties of the adsorbed structure on the silica substrate. This study will lead to an improved understanding of the dispersion mechanism and any changes in the adhesion properties of adsorbed micelle structures on the surface will lead to improved control of dispersion in nano particulate systems.

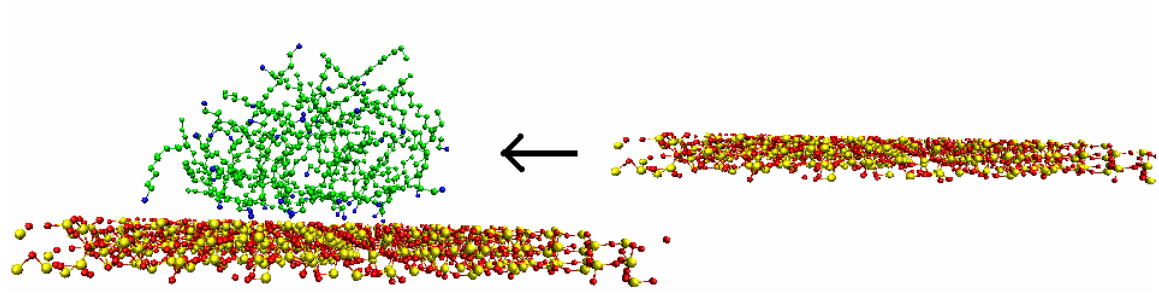


Figure 6-2: Lateral force measurement of adsorbed micellar structure on silica substrate.

Molecular level computer simulations such as those reported complement experimental methods by providing molecular level insight into the evolution of

surfactants and micelle in the bulk, the adsorption of surfactants at solid-liquid interfaces and properties of adsorbed structure at interfaces. This insight is expected to enhance the achievement of optimum performance in emerging and mature technologies such as controlled drug delivery, abrasives for precision polishing, coating, paints, and nano composite materials, which are increasingly relying on nano particulate materials which must be well-dispersed in solution with surfactants.

APPENDIX A
MICELLE CODE

```
/* This code is the latest and finally modified copy 2/19/05 */
/*
Filename - nozpbc.c
Language - C (gcc)
Date - 02/19/05
Original Author   - Kunal     Shah (kunal@ufl.edu)
Owner -Sinnott group at University of Florida
Description - This is a parallel version of     a sequential MD1 code
that simulates the motion     of particles (surfactants) in bulk
(water) and at silica and graphite surfaces.    The   code uses the
MPICH version     1.2.5 of the MPI standard.

Note: Modifications     made by     Mayank Jain are   commented and
marked under JAIN
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "mpi.h" /*     JAIN - This is the MPI library      */

double cosd(double); /* JAIN - This is the function   prototype of
'cosd'*/

#define SUR 48
#define CION 48
#define WT 3356
#define ASIZE 15000

main(int argc, char     **argv)
{
      MPI_Status status; /* JAIN - This is the MPI_Status   variable
used to establish inter-processor communications */
      int   myid; /* JAIN -This     is the id of the processor ranging
from  0 to total # of   processors */
      int   np;   /* JAIN     -This is the total number of processors
      */
      double st1, st2, st3, st4, st5, st6, st7, st8, st9; /* JAIN -
These variables are used to record start times */
      double et1, et2, et3, et4, et5, et6, et7, et8, et9; /* JAIN -
These variables are used to record end times */
      double tt1=0, tt2=0, tt3=0, tt4=0, tt5=0, tt6=0, tt7=0, tt8=0,
tt9=0; /* JAIN - These variables are used to record total (end - start)
times */
```

```
      FILE *fp,*fs,*fq,*fr;
      long lr,tstep,ne,it,nl,kp,kl,pq,gt,sr,st,xyz,index,count,plus;
      long
i,j,k,l,t,sel,g,q,d,e,f,time,ab,bc,cd,fg,mn,abc,temp,nk,xy,yz,p,lp,lg,l
n,pqr,cde,bcd,ind,kpr,str,pky,ps,ts,wt,skr,sqr,kqr;
      double
r,X,R,E,kr,ds,Ur,ar,br,cr,Fx,Fy,Fz,acc,delta,vt,x1new,y1new,z1new,vx1,v
y1,vz1;
        double
arx,bry,crz,rne,Rv,ex,a1x,b1y,c1z,a2x,b2y,c2z,a3x,b3y,c3z,r1,r2,r3,Fxw,
Fyw,Fzw;
      double
x,y,z,accx,accy,accz,ll,Fxs,Fys,Fzs,Rc,aq,Rw,Dw,bq,cq,dq,kb,mass;
      double
qi,qj,qij,Aii,Ajj,Aij,Cii,Cjj,Cij,Sij,Dij,Fxsur,Fysur,Fzsur,sumFxsur,su
mFysur,sumFzsur;
      double
sumforcex,sumforcey,sumforcez,sumFx,sumFy,sumFz,sumFxw,sumFyw,sumFzw,su
mFxsili,sumFysili,sumFzsili;
      double
Fx1,Fy1,Fz1,Fx2,Fy2,Fz2,Fx3,Fy3,Fz3,r12,r23,Fxsili,Fysili,Fzsili;
      double
aux,buy,cuz,rub,avx,bvy,cvz,rvb,awx,bwy,cwz,rwb,kw,thet,the;
      double Fxb,Fyb,Fzb,ex1,ex2,boxX,boxY,boxZ,rb0,theta;
      double Fxb1,Fyb1,Fzb1,Fxb2,Fyb2,Fzb2,Fxb3,Fyb3,Fzb3,Vb1,Vb2,Vb3;
      double
alj,blj,clj,rlj,Fxlj,Fylj,Fzlj,sumFxlj,sumFylj,sumFzlj,Aab,Bab,Cab,Dab,
qa,qb,eab,Sab,Ano,Cno;
      double Fxt,Fyt,Fzt,rt0,b0,a0,a1,rbond;
      double
sumFx12,sumFy12,sumFz12,Fx12,Fy12,Fz12,FRX,FRY,FRZ,FKX,FKY,FKZ,e0,Vol,m
odk,sumFKX,sumFKY,sumFKZ,sumVK;
      double
Vs,Vb,Vt,Vo,Vh,Voh1,Voh2,Vhh,Vlj,sumVlj,Vr,sumVr,V12,sumV12,PE,TPE=0.0,
PEnew,V,KEnew,PEsur,KEsur,EK,TEK=0.0,EKnew=0.0,PEL,TEKnew=0.0;

      double sinx,cosx,temp1,temp2;
      double rt01,rt02,rt03,rt04,rt05,rt06;

      double
r1r2Cx,r1r2Cy,r1r2Cz,modr1r2C,r2r3Cx,r2r3Cy,r2r3Cz,modr2r3C,phi,pi;
      double
Fxt11,Fyt11,Fzt11,Fxt22,Fyt22,Fzt22,Fxt33,Fyt33,Fzt33,Fxt44,Fyt44,Fzt44
;
      double Vt11,Vt22,Vt33,Vt44,a5t,b5t,c5t,r5t,a6t,b6t,c6t,r6t;
      double ar1,ar2,br1,br2,cr1,cr2,rs1,rs2;
      double
ab1,ab2,ab3,ab4,bb1,bb2,bb3,bb4,cb1,cb2,cb3,cb4,rb1,rb2,rb3,rb4,dotk,do
tk1,dotk2,dotk3,bend1,bend2,bend3,bend;
      double
a1t,b1t,c1t,r1t,a2t,b2t,c2t,r2t,a3t,b3t,c3t,r3t,a4t,b4t,c4t,r4t,Fxt1,Fy
t1,Fzt1,Fxt2,Fyt2,Fzt2;

      char ch,arr[5000],arra[5000];
```

```
      double
a[ASIZE],b[ASIZE],c[ASIZE],anew[ASIZE],bnew[ASIZE],cnew[ASIZE],a1new[AS
IZE],b1new[ASIZE],c1new[ASIZE];
       double
Accox[ASIZE],Accoy[ASIZE],Accoz[ASIZE],Velox[ASIZE],Veloy[ASIZE],Veloz[
ASIZE];

      double acclx[2][ASIZE],accly[2][ASIZE],acclz[2][ASIZE];
      double
Accoxnew[ASIZE],Accoynew[ASIZE],Accoznew[ASIZE],Veloxnew[ASIZE],Veloyne
w[ASIZE],Veloznew[ASIZE];
      long no[ASIZE],sele[ASIZE],lange[ASIZE],neb[ASIZE];

      double
vx[2][ASIZE],vy[2][ASIZE],vz[2][ASIZE],Vel[2][ASIZE],T[2][ASIZE],KE,TKE
=0.0;

      int
itt,nonew[ASIZE],selenew[ASIZE],langenew[ASIZE],iii,isur,icion,iwt,tsur
,tcion,twt;

      long **neighbor;
      int *mod,*modsur,*modcion,*modwt;
      int size2;

      double akx,bky,ckz,rke,rad;

      long MAXK,KMAX,KSQMAX,KSQ,KX,KY,KZ,TOTK;
      double KAPPA,TWOPI,B,RKX,RKY,RKZ,RKSQ,KVEC[1000],RSQPI;
      double A1,A2,A3,A4,A5,P,TE,KRIJ,KRIJSQ,TEP,ERFC;

      double
ImagX[50],RealX[50],ImagY[50],RealY[50],ImagZ[50],RealZ[50],RealR,ImagR
;
      double
real,imag,RealSum,ImagSum,realtemp,imagtemp,VS,VK,VD,FACTOR;

      MPI_Init(&argc,&argv); /* JAIN - Initialize     the   MPI
      system and create command line arguments */
      MPI_Comm_size(MPI_COMM_WORLD,&np); /* JAIN - Initialize     the
      total number of   processors*/
      MPI_Comm_rank(MPI_COMM_WORLD,&myid); /*   JAIN - Initilaize the
processor   id and ranks*/

      if(myid==0)     /* JAIN - If master     processor then enter if
loop */
                printf("Number of Processors = %d \n",np); /* JAIN -
Print the total number     of processors */

        st1 = MPI_Wtime();
        st8 = MPI_Wtime();


      Rc=8.0;
      boxX=59.0;
      boxY=61.0;
      boxZ=75.0;
```

```
Vol=boxX*boxY*boxZ;
e0=0.00553;
MAXK=1000;
KMAX=5;
KSQMAX=18;
TWOPI=6.2832;
KAPPA=3.2/Rc;
RSQPI=0.5642;
B=1.0/(4.0*KAPPA*KAPPA);
TOTK=0;
A1=0.25483;
A2=-0.28449;
A3=1.42141;
A4=-1.45315;
A5=1.061405;
P=0.32759;

ab=bc=cd=fg=mn=0;
d=e=f=yz=lp=ln=0;
aq=58.24;
Rw=2.361;
Dw=4.4195;
bq=11.2543;
cq=-9.1698;
dq=4.844;
ps=ind=wt=0;
kb=1.34;
rad=(22.0/(7.0*180.0));
theta=112*rad;
eab=0.00722;
Sab=3.364;
index=0;
rb0=5.3858;
b0=1.539;
a0=0.1115;
a1=0.6669;
rbond=39.02;
count=0;
pi=22.0/7.0;
rt01=0.09599;
  rt02=0.12576;
  rt03=0.135739;
  rt04=0.031655;
  rt05=0.271479;
  rt06=0.32584;


fp=fopen("nxsc220","r");
fs=fopen("nxso230","w");
fq=fopen("nxse230","w");
fr=fopen("nxsc230","w");
if(fp==NULL)
/*check of opening files*/
{
      puts("cannot open file");
      return (1);
}
```

```
if(fs==NULL)
{
      puts("cannot open file");
      return (1);
}
if(fq==NULL)
{
      puts("cannot open file");
      return (1);
}
if(fr==NULL)
{
      puts("cannot open file");
      return (1);
}

for(lr=0;lr<9;lr++)
/*Reading first line of input file*/
{
      xy=0;
      while(1)
      {
            ch=fgetc(fp);
            if(ch==','||ch==' '||ch=='\n')
                  break;
            arra[xy]=ch;
            xy++;
      }
      arra[xy]='\0';
      if(lr==0)
            nk=atoi(arra);
      else
      if(lr==2)
            time=atoi(arra);
      else
      if(lr==5)
            temp=atoi(arra);
      else
      if(lr==8)
            delta=atof(arra);

}
for( t=0;t<nk;t++)
/*Reading entire input file from second line*/
{
      for( i=0;i<12;i++)
      {
            j=0;

            while(1)
            {
                  ch=fgetc(fp);
                  if(ch==','||ch==' '||ch=='\n')
                        break;
                  arr[j]=ch;
                  j++;
            }
```

```
            arr[j]='\0';
            if(i==0)
            {
                  p=atoi(arr);
                  no[yz]=p;
                  yz++;
            }
            if(i==3)
            {
                  ll=atoi(arr);
                  sele[lp]=ll;
                  lp++;
            }
            if(i==6)
            {
                  x=atof(arr);
                  a[d]=x;
                  d++;
            }
            if(i==7)
            {
                  y=atof(arr);
                  b[e]=y;
                  e++;
            }
            if(i==8)
            {
                  z=atof(arr);
                  c[f]=z;
                  f++;
            }
            if(i==11)
            {
                  lg=atoi(arr);
                  lange[ln]=lg;
                  ln++;
            }
      }
}

 neighbor=malloc(((nk/np)+np)*sizeof(long *));
/*Dynamic memory allocation of arrays*/
   for(i=0;i<(nk/np)+np;i++)
   {
          neighbor[i]=malloc(150*sizeof(long));
   }
   for(i=0;i<(nk/np)+np;i++)
   {
          for(j=0;j<150;j++)
          {
                neighbor[i][j]=0;
          }
   }

  mod=malloc(np*sizeof(int));
modsur=malloc(np*sizeof(int));
modcion=malloc(np*sizeof(int));
```

```
      modwt=malloc(np*sizeof(int));

         for(i=0;i<np;i++)
         {
                mod[i]=0;
            modsur[i]=0;
            modcion[i]=0;
            modwt[i]=0;
         }

       if(nk%np!=0)
            mod[np-1]=(nk%np);
       if(SUR%np!=0)
      {
            modsur[np-1]=(SUR%np);

      }

      if(CION%np!=0)
        {
                modcion[np-1]=(CION%np);
        }


       if(WT%np!=0)
        {
                modwt[np-1]=(WT%np);
        }

                                                  /*Modified Atom
Decomposition method-- Dividing equally all the different particles on
each processors*/
/*

      isur=SUR/np;
        icion=CION/np;
        iwt=WT/np;
        tsur=0;
        tcion=0;
        twt=0;

        itt=0;

        if(myid<=np)
        {
                for(iii=1;iii<np+1;iii++)
                {
                        for(it=0;it<nk;it++)
                        {
                                if(sele[it]==4||sele[it]==1)
                                {

if(lange[it]<=((iii*isur)+modsur[iii-1])&&lange[it]>tsur)
                                                  {


nonew[itt]=itt+1;
```

```
selenew[itt]=sele[it];

anew[itt]=a[it];

bnew[itt]=b[it];

cnew[itt]=c[it];

langenew[itt]=(lange[it]-(iii-1)*isur)+(iii-1)*(isur+icion+iwt);

                                                        itt++;
                                                }
                                        }
                                        else
                                        if(sele[it]==7)
                                        {

if(lange[it]<=((48+iii*icion)+modcion[iii-1])&&lange[it]>tcion)
                                                {


nonew[itt]=itt+1;

selenew[itt]=sele[it];

anew[itt]=a[it];

bnew[itt]=b[it];

cnew[itt]=c[it];

langenew[itt]=lange[it]-SUR+isur+modsur[iii-1]-(iii-1)*icion+(iii-
1)*(isur+icion+iwt);

                                                        itt++;
                                                }
                                        }
                                else

if(sele[it]==3||sele[it]==2||sele[it]==5)
                                        {

if(lange[it]<=((96+(iii*iwt))+modwt[iii-1])&&lange[it]>twt)
                                                {

nonew[itt]=itt+1;

selenew[itt]=sele[it];

anew[itt]=a[it];

bnew[itt]=b[it];

cnew[itt]=c[it];
```

```
langenew[itt]=lange[it]-(SUR+CION)+isur+modsur[iii-
1]+icion+modcion[iii-1]-(iii-1)*iwt+(iii-1)*(isur+icion+iwt);

                                                        itt++;
                                                }
                                        }
                                }
                                tsur=iii*isur;
                                tcion=SUR+iii*icion;
                                twt=(SUR+CION)+iii*iwt;
                        }

                        for(it=0;it<nk;it++)
                        {
                                no[it]=nonew[it];
                                sele[it]=selenew[it];
                                a[it]=anew[it];
                                b[it]=bnew[it];
                                c[it]=cnew[it];
                                lange[it]=langenew[it];
                        }
                        for(it=0;it<nk;it++)
                        {
                                 printf("%d   %d    %lf    %lf    %lf
%d\n",no[it],sele[it],a[it],b[it],c[it],lange[it]);
                        }

                }
                if(myid==5)
                {
                         for(it=0;it<nk;it++)
                        {
                                 printf("%d   %d    %lf    %lf    %lf
%d\n",no[it],sele[it],a[it],b[it],c[it],lange[it]);
                        }
                }*/


              for(KX=0;KX<=KMAX;KX++)
              /*Ewald summation*/
              {
                    RKX=TWOPI*KX;
                    for(KY=-KMAX;KY<=KMAX;KY++)
                    {
                          RKY=TWOPI*KY;
                          for(KZ=-KMAX;KZ<=KMAX;KZ++)
                          {
                                RKZ=TWOPI*KZ;
                                KSQ=KX*KX+KY*KY+KZ*KZ;

                                if((KSQ<KSQMAX)&&(KSQ!=0))
                                {
                                      TOTK=TOTK+1;

                                      if(TOTK>=MAXK)
                                            continue;
```

```
                              RKSQ=RKX*RKX+RKY*RKY+RKZ*RKZ;
                              KVEC[TOTK]=exp(-B*RKSQ)/RKSQ;
                        }
                  }
            }
      }

      et1 = MPI_Wtime();
      tt1 = et1 - st1;
      if (myid == 0)
        {
          printf("Time to initialize (open and read file etc.): %f \n",
tt1);
        }


      for( tstep=0;tstep<time;tstep++)
      /*First main loop- time step loop*/
      {
            PE=0.0;
            KE=0.0;
            PEsur=0.0;
            KEsur=0.0;
            PEL=0.0;

            if(tstep==1000*index)
      /*Print statements of coordinates of chosen particles- 'o'utput
file*/
            {
                  index++;

                  fprintf(fs," %d          3           0
0\n",nk);
                  fprintf(fs,"\n");

                  for( it=0;it<nk;it++)
                  {
                        if(sele[it]==1)
                              fprintf(fs,"  7     %30.15lf   %30.15lf
      %30.15lf\n",a[it],b[it],c[it]);
                        else

      if(sele[it]==4||sele[it]==6||sele[it]==11||sele[it]==14)
                                    fprintf(fs,"  6     %30.15lf
%30.15lf    %30.15lf\n",a[it],b[it],c[it]);

      /*                return
                        if(sele[it]==2||sele[it]==5||sele[it]==13)
                        fprintf(fs,"  1     %30.15lf   %30.15lf
      %30.15lf\n",a[it],b[it],c[it]);
                        else
                        if(sele[it]==3||sele[it]==12||sele[it]==16)
                        fprintf(fs,"  8     %30.15lf   %30.15lf
      %30.15lf\n",a[it],b[it],c[it]);
      */
                        else
```

```
                        if(sele[it]==7)
                              fprintf(fs,"  9     %30.15lf   %30.15lf
      %30.15lf\n",a[it],b[it],c[it]);
                        else
                                if(sele[it]==25)
                                        fprintf(fs,"  14   %30.15lf
%30.15lf       %30.15lf\n",a[it],b[it],c[it]);
                                else
                                if(sele[it]==50)
                                        fprintf(fs,"  8   %30.15lf
%30.15lf       %30.15lf\n",a[it],b[it],c[it]);
                           }
            }
            if(tstep==0)
      /*Restarting the job- Reading velocities of previous step of all
the particles from the input file*/
            {
                  for( t=0;t<nk;t++)
                  {
                        for( i=0;i<12;i++)
                        {
                              j=0;

                              while(1)
                              {
                                    ch=fgetc(fp);
                                    if(ch==','||ch==' '||ch=='\n')
                                          break;
                                    arr[j]=ch;
                                    j++;
                              }
                              arr[j]='\0';
                              if(i==0)
                              {
                                    no[t]=atoi(arr);
                              }
                              if(i==3)
                              {
                                    sele[t]=atoi(arr);
                              }
                              if(i==6)
                              {
                                    vx[ps][t]=atof(arr);
                              }
                              if(i==7)
                              {
                                    vy[ps][t]=atof(arr);
                              }
                              if(i==8)
                              {
                                    vz[ps][t]=atof(arr);
                              }
                              if(i==11)
                              {
                                    lange[t]=atoi(arr);
                              }
                        }
```

```
                  }
            }

            if(tstep==5*ind)
      /*Making the neighbor list for all the particles*/
            {
                  ind++;

                  for(
ne=myid*(nk/np);ne<(myid*(nk/np)+(nk/np)+mod[myid]);ne++)
                  {
                        kpr=0;

                        for( nl=0;nl<nk;nl++)
                       {
                              if(ne==nl)
                                    continue;
                              arx=a[ne]-a[nl];
                              bry=b[ne]-b[nl];
                              crz=c[ne]-c[nl];

                              if(arx>(0.5*boxX))
                                    arx=boxX-arx;
                              else
                                    if(arx<-(0.5*boxX))
                                          arx=boxX+arx;

                              if(bry>(0.5*boxY))
                                    bry=bry-boxY;
                              else
                              if(bry<-(0.5*boxY))
                                    bry=boxY+bry;

                              if(crz>(0.5*boxZ))
                                    crz=crz-boxZ;
                              else
                              if(crz<-(0.5*boxZ))
                                    crz=boxZ+crz;


      rne=pow(((arx*arx)+(bry*bry)+(crz*crz)),0.5);


                              if(rne<Rc)
                              {
                                    size2=ne-(myid*(nk/np));
                                    neighbor[size2][kpr]=nl;

                                    /*if(myid==1)
                                          printf("%d     %d     %d
%d      %d      %d      %d      %d\n",ind,myid,ne,ne-(myid*(nk/np)),nl,
neighbor[size2][kpr],size2,kpr);
                                    */
                              kpr++;

                              }
```

```
                        }
                        neb[ne]=kpr;
                        /*printf("%d
neighbor=%d\n",tstep,kpr);*/
                  }
            }

            if(ps==2)
            {
                  ps=ps-2;
            }

            /* JAIN     - Begin     Modifications*/

            /* JAIN     - This is the second 'for loop'     that goes
from 1 to   total #     of particles (nk). In this parallel version,
each processor    is assigned nk/np particles. So, the first processor
computes this 'for      loop' for the first     set   of nk/np
particles,  the   second processor for the second     set   of nk/np
particles and so on  */

            for(
pq=myid*(nk/np);pq<(myid*(nk/np)+(nk/np)+mod[myid]);pq++)

                  /* JAIN     - End of Modifications*/
            {
                  sumVr=0.0;
                  sumV12=0.0;
                  sumVlj=0.0;
                  Vo=Vh=Vb=0.0;
                  Vt=Vs=0.0;
                  sumforcex=sumforcey=sumforcez=0.0;
                  sumFx12=sumFy12=sumFz12=0.0;
                  sumFxlj=sumFylj=sumFzlj=0.0;
                  sumFxw=sumFyw=sumFzw=0.0;
                  sumFxsur=sumFysur=sumFzsur=0.0;
                  sumFxsili=sumFysili=sumFzsili=0.0;
                  Fxb=Fyb=Fzb=0.0;
                  Fxb1=Fyb1=Fzb1=0.0;
                  Fxb2=Fyb2=Fzb2=0.0;
                  Fxb3=Fyb3=Fzb3=0.0;
                  Vb1=Vb2=Vb3=0.0;
                  Fxt=Fyt=Fzt=0.0;
                  Fxs=Fys=Fzs=0.0;
                  rb1=rb2=rb3=rb4=0.0;
                  ab1=bb1=cb1=ab2=bb2=cb2=ab3=bb3=cb3=ab4=bb4=cb4=0.0;
                  sumFKX=sumFKY=sumFKZ=0.0;
                  sumVK=0.0;
                  Fxt1=Fyt1=Fzt1=Fxt2=Fyt2=Fzt2=0.0;
                  Fxt11=Fyt11=Fzt11=Fxt22=Fyt22=Fzt22=0.0;
                  Fxt33=Fyt33=Fzt33=Fxt44=Fyt44=Fzt44=0.0;
                  Vt11=Vt22=Vt33=Vt44=0.0;

      /*Assigning masses to each particle*/
                  if(sele[pq]==1)
```

```
                        mass=74.0;
                  if(sele[pq]==2||sele[pq]==5)
                        mass=1.0;
                  if(sele[pq]==3)
                        mass=16.0;
                  if(sele[pq]==4)
                        mass=14.0;
                  if(sele[pq]==6)
                        mass=12.0;
                  if(sele[pq]==7)
                        mass=79.91;
                  if(sele[pq]==25)
                        mass=28.0855;
                  if(sele[pq]==50)
                        mass=16.0;


                  if(sele[pq]==4||sele[pq]==1)
                  {
                  /*Bending, Torsion & Stretching potentials from
Weber, T. A. (1978). J. Chem.Phys 69: 2347-2354.and DL_POLY manual*/


      /*Bending*/
                        if(pq!=(nk-1))
                        {
                              ab3=a[pq]-a[pq+1];
                              bb3=b[pq]-b[pq+1];
                              cb3=c[pq]-c[pq+1];

                              if(ab3>(0.5*boxX))
                                    ab3=ab3-boxX;
                              else
                              if(ab3<-(0.5*boxX))
                                    ab3=boxX+ab3;

                              if(bb3>(0.5*boxY))
                                    bb3=bb3-boxY;
                              else
                              if(bb3<-(0.5*boxY))
                                    bb3=boxY+bb3;

                              if(cb3>(0.5*boxZ))
                                    cb3=cb3-boxZ;
                              else
                              if(cb3<-(0.5*boxZ))
                                    cb3=boxZ+cb3;


      rb3=pow(((ab3*ab3)+(bb3*bb3)+(cb3*cb3)),0.5);
                        }
                        if(pq!=0)
                        {
                              ab1=a[pq]-a[pq-1];
                              bb1=b[pq]-b[pq-1];
                              cb1=c[pq]-c[pq-1];
```

```
                        if(ab1>(0.5*boxX))
                              ab1=ab1-boxX;
                        else
                        if(ab1<-(0.5*boxX))
                              ab1=boxX+ab1;

                        if(bb1>(0.5*boxY))
                              bb1=bb1-boxY;
                        else
                        if(bb1<-(0.5*boxY))
                              bb1=boxY+bb1;

                        if(cb1>(0.5*boxZ))
                              cb1=cb1-boxZ;
                        else
                        if(cb1<-(0.5*boxZ))
                              cb1=boxZ+cb1;


      rb1=pow(((ab1*ab1)+(bb1*bb1)+(cb1*cb1)),0.5);
                        }
                        if(pq!=(nk-1)&&pq!=(nk-2))
                        {
                              ab2=a[pq+1]-a[pq+2];
                                    bb2=b[pq+1]-b[pq+2];
                                    cb2=c[pq+1]-c[pq+2];

                                    if(ab2>(0.5*boxX))
                                          ab2=ab2-boxX;
                                    else
                                    if(ab2<-(0.5*boxX))
                                          ab2=boxX+ab2;

                                    if(bb2>(0.5*boxY))
                                          bb2=bb2-boxY;
                                    else
                                    if(bb2<-(0.5*boxY))
                                          bb2=boxY+bb2;

                                   if(cb2>(0.5*boxZ))
                                          cb2=cb2-boxZ;
                                    else
                                    if(cb2<-(0.5*boxZ))
                                          cb2=boxZ+cb2;


    rb2=pow(((ab2*ab2)+(bb2*bb2)+(cb2*cb2)),0.5);
                        }
                        if(pq!=0&&pq!=1)
                        {
                              ab4=a[pq-1]-a[pq-2];
                                    bb4=b[pq-1]-b[pq-2];
                                    cb4=c[pq-1]-c[pq-2];

                                    if(ab4>(0.5*boxX))
                                          ab4=ab4-boxX;
                                    else
```

```
                                        if(ab4<-(0.5*boxX))
                                                ab4=boxX+ab4;

                                        if(bb4>(0.5*boxY))
                                                bb4=bb4-boxY;
                                        else
                                        if(bb4<-(0.5*boxY))
                                                bb4=boxY+bb4;

                                        if(cb4>(0.5*boxZ))
                                                cb4=cb4-boxZ;
                                        else
                                        if(cb4<-(0.5*boxZ))
                                                cb4=boxZ+cb4;


rb4=pow(((ab4*ab4)+(bb4*bb4)+(cb4*cb4)),0.5);
                        }

                        if(pq==0||lange[pq-1]!=lange[pq])
                        {
                              dotk2=((ab3*(-ab2))+(bb3*(-bb2))+(cb3*(-
cb2)))/(rb2*rb3);

                                        bend2=acos(dotk2);

                                        Vb2=0.5*rb0*pow((bend2-
theta),2.0);

                                        Fxb2=(rb0/sin(bend2))*(bend2-
theta)*((-ab2)/(rb2*rb3)-cos(bend2)*(ab3/(rb3*rb3)));
                                        Fyb2=(rb0/sin(bend2))*(bend2-
theta)*((-bb2)/(rb2*rb3)-cos(bend2)*(bb3/(rb3*rb3)));
                                        Fzb2=(rb0/sin(bend2))*(bend2-
theta)*((-cb2)/(rb2*rb3)-cos(bend2)*(cb3/(rb3*rb3)));

                                Vb=Vb2;

                                        Fxb=Fxb2;
                                        Fyb=Fyb2;
                                        Fzb=Fzb2;

                        }
                        else
                        if(pq==(nk-1)||lange[pq+1]!=lange[pq])
                        {
                              dotk3=(((-ab4)*ab1)+((-bb4)*bb1)+((-
cb4)*cb1))/(rb1*rb4);

                                        bend3=acos(dotk3);

                                        Vb3=0.5*rb0*pow((bend3-
theta),2.0);


                                        Fxb3=(rb0/sin(bend3))*(bend3-
theta)*((-ab4)/(rb1*rb4)-cos(bend3)*(ab1/(rb1*rb1)));
```

```
                                        Fyb3=(rb0/sin(bend3))*(bend3-
theta)*((-bb4)/(rb1*rb4)-cos(bend3)*(bb1/(rb1*rb1)));
                                        Fzb3=(rb0/sin(bend3))*(bend3-
theta)*((-cb4)/(rb1*rb4)-cos(bend3)*(cb1/(rb1*rb1)));

                              Vb=Vb3;

                                        Fxb=Fxb3;
                                        Fyb=Fyb3;
                                        Fzb=Fzb3;

                         }
                         else
                         if(pq==(nk-
2)||(lange[pq+2]!=lange[pq]&&lange[pq+1]==lange[pq]))
                         {

      dotk1=((ab3*ab1)+(bb3*bb1)+(cb3*cb1))/(rb1*rb3);

                                        bend1=acos(dotk1);


                                        Vb1=0.5*rb0*pow((bend1-
theta),2.0);

                                        Fxb1=(rb0/sin(bend1))*(bend1-
theta)*(ab3/(rb1*rb3)+ab1/(rb1*rb3)-
cos(bend1)*(ab3/(rb3*rb3)+ab1/(rb1*rb1)));
                                        Fyb1=(rb0/sin(bend1))*(bend1-
theta)*(bb3/(rb1*rb3)+bb1/(rb1*rb3)-
cos(bend1)*(bb3/(rb3*rb3)+bb1/(rb1*rb1)));
                                        Fzb1=(rb0/sin(bend1))*(bend1-
theta)*(cb3/(rb1*rb3)+cb1/(rb1*rb3)-
cos(bend1)*(cb3/(rb3*rb3)+cb1/(rb1*rb1)));


                              dotk3=(((-ab4)*ab1)+((-bb4)*bb1)+((-
cb4)*cb1))/(rb1*rb4);

                                        bend3=acos(dotk3);

                                        Vb3=0.5*rb0*pow((bend3-
theta),2.0);


                                        Fxb3=(rb0/sin(bend3))*(bend3-
theta)*((-ab4)/(rb1*rb4)-cos(bend3)*(ab1/(rb1*rb1)));
                                        Fyb3=(rb0/sin(bend3))*(bend3-
theta)*((-bb4)/(rb1*rb4)-cos(bend3)*(bb1/(rb1*rb1)));
                                        Fzb3=(rb0/sin(bend3))*(bend3-
theta)*((-cb4)/(rb1*rb4)-cos(bend3)*(cb1/(rb1*rb1)));

                              Vb=Vb1+Vb3;

                                        Fxb=Fxb1+Fxb3;
                                        Fyb=Fyb1+Fyb3;
                                        Fzb=Fzb1+Fzb3;
```

```
                        }
                        else
                        if(pq==1||(lange[pq-2]!=lange[pq]&&lange[pq-
1]==lange[pq]))
                        {

      dotk1=((ab3*ab1)+(bb3*bb1)+(cb3*cb1))/(rb1*rb3);

                                        bend1=acos(dotk1);


                                        Vb1=0.5*rb0*pow((bend1-
theta),2.0);

                                        Fxb1=(rb0/sin(bend1))*(bend1-
theta)*(ab3/(rb1*rb3)+ab1/(rb1*rb3)-
cos(bend1)*(ab3/(rb3*rb3)+ab1/(rb1*rb1)));
                                        Fyb1=(rb0/sin(bend1))*(bend1-
theta)*(bb3/(rb1*rb3)+bb1/(rb1*rb3)-
cos(bend1)*(bb3/(rb3*rb3)+bb1/(rb1*rb1)));
                                        Fzb1=(rb0/sin(bend1))*(bend1-
theta)*(cb3/(rb1*rb3)+cb1/(rb1*rb3)-
cos(bend1)*(cb3/(rb3*rb3)+cb1/(rb1*rb1)));


                                        dotk2=((ab3*(-ab2))+(bb3*(-
bb2))+(cb3*(-cb2)))/(rb2*rb3);

                                        bend2=acos(dotk2);

                                        Vb2=0.5*rb0*pow((bend2-
theta),2.0);

                                        Fxb2=(rb0/sin(bend2))*(bend2-
theta)*((-ab2)/(rb2*rb3)-cos(bend2)*(ab3/(rb3*rb3)));
                                        Fyb2=(rb0/sin(bend2))*(bend2-
theta)*((-bb2)/(rb2*rb3)-cos(bend2)*(bb3/(rb3*rb3)));
                                        Fzb2=(rb0/sin(bend2))*(bend2-
theta)*((-cb2)/(rb2*rb3)-cos(bend2)*(cb3/(rb3*rb3)));

                                Vb=Vb1+Vb2;

                                        Fxb=Fxb1+Fxb2;
                                        Fyb=Fyb1+Fyb2;
                                        Fzb=Fzb1+Fzb2;
                        }
                        else
                                {

dotk1=((ab3*ab1)+(bb3*bb1)+(cb3*cb1))/(rb1*rb3);

                                        bend1=acos(dotk1);


                                        Vb1=0.5*rb0*pow((bend1-
theta),2.0);
```

```
                              Fxb1=(rb0/sin(bend1))*(bend1-
theta)*(ab3/(rb1*rb3)+ab1/(rb1*rb3)-
cos(bend1)*(ab3/(rb3*rb3)+ab1/(rb1*rb1)));
                              Fyb1=(rb0/sin(bend1))*(bend1-
theta)*(bb3/(rb1*rb3)+bb1/(rb1*rb3)-
cos(bend1)*(bb3/(rb3*rb3)+bb1/(rb1*rb1)));
                              Fzb1=(rb0/sin(bend1))*(bend1-
theta)*(cb3/(rb1*rb3)+cb1/(rb1*rb3)-
cos(bend1)*(cb3/(rb3*rb3)+cb1/(rb1*rb1)));



                                        dotk2=((ab3*(-ab2))+(bb3*(-
bb2))+(cb3*(-cb2)))/(rb2*rb3);

                                        bend2=acos(dotk2);

                                        Vb2=0.5*rb0*pow((bend2-
theta),2.0);

                                        Fxb2=(rb0/sin(bend2))*(bend2-
theta)*((-ab2)/(rb2*rb3)-cos(bend2)*(ab3/(rb3*rb3)));
                                        Fyb2=(rb0/sin(bend2))*(bend2-
theta)*((-bb2)/(rb2*rb3)-cos(bend2)*(bb3/(rb3*rb3)));
                                        Fzb2=(rb0/sin(bend2))*(bend2-
theta)*((-cb2)/(rb2*rb3)-cos(bend2)*(cb3/(rb3*rb3)));


                                        dotk3=(((-ab4)*ab1)+((-
bb4)*bb1)+((-cb4)*cb1))/(rb1*rb4);

                                        bend3=acos(dotk3);

                                        Vb3=0.5*rb0*pow((bend3-
theta),2.0);


                                        Fxb3=(rb0/sin(bend3))*(bend3-
theta)*((-ab4)/(rb1*rb4)-cos(bend3)*(ab1/(rb1*rb1)));
                                        Fyb3=(rb0/sin(bend3))*(bend3-
theta)*((-bb4)/(rb1*rb4)-cos(bend3)*(bb1/(rb1*rb1)));
                                        Fzb3=(rb0/sin(bend3))*(bend3-
theta)*((-cb4)/(rb1*rb4)-cos(bend3)*(cb1/(rb1*rb1)));

                              Vb=Vb1+Vb2+Vb3;

                              Fxb=Fxb1+Fxb2+Fxb3;
                              Fyb=Fyb1+Fyb2+Fyb3;
                              Fzb=Fzb1+Fzb2+Fzb3;
                         }

                              /*Torsional*/

                         if(pq==0||lange[pq-1]!=lange[pq])
                                 {
```

```
                                        a1t=a[pq]-a[pq+1];
                /*1st loop*/
                                        b1t=b[pq]-b[pq+1];
                                        c1t=c[pq]-c[pq+1];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq+1]-a[pq+2];
                                        b2t=b[pq+1]-b[pq+2];
                                        c2t=c[pq+1]-c[pq+2];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq+2]-a[pq+3];
                                        b3t=b[pq+2]-b[pq+3];
                                        c3t=c[pq+2]-c[pq+3];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
```

```
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt11=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));


                                        Fxt1=(-
a2t*(b2t*b3t+c2t*c3t)+a3t*(b2t*b2t+c2t*c2t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b2t*(a2t*a3t+c2t*c3t)+b3t*(a2t*a2t+c2t*c2t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c2t*(b2t*b3t+a2t*a3t)+c3t*(b2t*b2t+a2t*a2t))/(modr1r2C*modr2r3C);

                                        Fxt2=-0.5*cos(phi)*((-
2.0*a1t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr1r2C
));
                                        Fyt2=-0.5*cos(phi)*((-
2.0*b1t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr1r2C
));
```

```
                                        Fzt2=-0.5*cos(phi)*((-
2.0*c1t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr1r2C
));


                                        Fxt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                                Vt=Vt11;

                                        Fxt=Fxt11;
                                        Fyt=Fyt11;
                                        Fzt=Fzt11;
                                  }
                           else
                                  if(pq==1||(lange[pq-
2]!=lange[pq]&&lange[pq-1]==lange[pq]))
                                  {
                                a1t=a[pq]-a[pq+1];
/*1st loop*/
                                        b1t=b[pq]-b[pq+1];
                                        c1t=c[pq]-c[pq+1];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq+1]-a[pq+2];
                                        b2t=b[pq+1]-b[pq+2];
                                        c2t=c[pq+1]-c[pq+2];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
```

```
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq+2]-a[pq+3];
                                        b3t=b[pq+2]-b[pq+3];
                                        c3t=c[pq+2]-c[pq+3];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);
```

```
phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt11=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));


                                        Fxt1=(-
a2t*(b2t*b3t+c2t*c3t)+a3t*(b2t*b2t+c2t*c2t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b2t*(a2t*a3t+c2t*c3t)+b3t*(a2t*a2t+c2t*c2t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c2t*(b2t*b3t+a2t*a3t)+c3t*(b2t*b2t+a2t*a2t))/(modr1r2C*modr2r3C);

                                        Fxt2=-0.5*cos(phi)*((-
2.0*a1t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr1r2C
));
                                        Fyt2=-0.5*cos(phi)*((-
2.0*b1t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr1r2C
));
                                        Fzt2=-0.5*cos(phi)*((-
2.0*c1t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr1r2C
));


                                        Fxt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);


                                        a1t=a[pq-1]-a[pq];
            /*2nd loop*/
                                        b1t=b[pq-1]-b[pq];
                                        c1t=c[pq-1]-c[pq];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
```

```
                                        c1t=c1t-boxZ;
                                else
                                if(c1t<-(0.5*boxZ))
                                        c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                a2t=a[pq]-a[pq+1];
                                b2t=b[pq]-b[pq+1];
                                c2t=c[pq]-c[pq+1];

                                if(a2t>(0.5*boxX))
                                        a2t=a2t-boxX;
                                else
                                if(a2t<-(0.5*boxX))
                                        a2t=boxX+a2t;

                                if(b2t>(0.5*boxY))
                                        b2t=b2t-boxY;
                                else
                                if(b2t<-(0.5*boxY))
                                        b2t=boxY+b2t;

                                if(c2t>(0.5*boxZ))
                                        c2t=c2t-boxZ;
                                else
                                if(c2t<-(0.5*boxZ))
                                        c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                            a3t=a[pq+1]-a[pq+2];
                                b3t=b[pq+1]-b[pq+2];
                                c3t=c[pq+1]-c[pq+2];

                                if(a3t>(0.5*boxX))
                                        a3t=a3t-boxX;
                                else
                                if(a3t<-(0.5*boxX))
                                        a3t=boxX+a3t;

                                if(b3t>(0.5*boxY))
                                        b3t=b3t-boxY;
                                else
                                if(b3t<-(0.5*boxY))
                                        b3t=boxY+b3t;

                                if(c3t>(0.5*boxZ))
                                        c3t=c3t-boxZ;
                                else
                                if(c3t<-(0.5*boxZ))
                                        c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);
```

```
                                r1r2Cx=b1t*c2t-b2t*c1t;
                                          r1r2Cy=-(a1t*c2t-a2t*c1t);
                                          r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                          r2r3Cx=b2t*c3t-b3t*c2t;
                                          r2r3Cy=-(a2t*c3t-a3t*c2t);
                                          r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                Vt22=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));
                                          Fxt1=(-
a1t*(b2t*b3t+c2t*c3t)+a2t*(b2t*b3t+c2t*c3t)-a3t*(b1t*b2t+c1t*c2t)-
a3t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b1t*b3t+c1t*c3t))/(modr1r2C*modr2r3C);
                                          Fyt1=(-
b1t*(a2t*a3t+c2t*c3t)+b2t*(a2t*a3t+c2t*c3t)-b3t*(a1t*a2t+c1t*c2t)-
b3t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a1t*a3t+c1t*c3t))/(modr1r2C*modr2r3C);
                                          Fzt1=(-
c1t*(b2t*b3t+a2t*a3t)+c2t*(b2t*b3t+a2t*a3t)-c3t*(b1t*b2t+a1t*a2t)-
c3t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b1t*b3t+a1t*a3t))/(modr1r2C*modr2r3C);

Fxt2=-
0.5*cos(phi)*(((2.0*a1t*(b2t*b2t+c2t*c2t)+2.0*a1t*(b1t*b2t+c1t*c2t)-
2.0*a2t*(b1t*b1t+c1t*c1t)-
2.0*a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr1r2C))+((2.0*a3t*(b2t*b3t+c2t*
c3t)-2.0*a2t*(b3t*b3t+c3t*c3t))/(modr2r3C*modr2r3C)));

Fyt2=-
0.5*cos(phi)*(((2.0*b1t*(a2t*a2t+c2t*c2t)+2.0*b1t*(a1t*a2t+c1t*c2t)-
2.0*b2t*(a1t*a1t+c1t*c1t)-
2.0*b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr1r2C))+((2.0*b3t*(a2t*a3t+c2t*
c3t)-2.0*b2t*(a3t*a3t+c3t*c3t))/(modr2r3C*modr2r3C)));

Fzt2=-
0.5*cos(phi)*(((2.0*c1t*(b2t*b2t+a2t*a2t)+2.0*c1t*(b1t*b2t+a1t*a2t)-
2.0*c2t*(b1t*b1t+a1t*a1t)-
2.0*c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr1r2C))+((2.0*c3t*(b2t*b3t+a2t*
a3t)-2.0*c2t*(b3t*b3t+a3t*a3t))/(modr2r3C*modr2r3C)));


                                          Fxt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                          Fyt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
```

```
                                        Fzt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                                Vt=Vt11+Vt22;

                                        Fxt=Fxt11+Fxt22;
                                        Fyt=Fyt11+Fyt22;
                                        Fzt=Fzt11+Fzt22;

                                }
                            else
                                  if(pq==2||(lange[pq-
3]!=lange[pq]&&lange[pq-2]==lange[pq]))
                                  {
                                a1t=a[pq]-a[pq+1];
/*1st loop*/
                                        b1t=b[pq]-b[pq+1];
                                        c1t=c[pq]-c[pq+1];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq+1]-a[pq+2];
                                        b2t=b[pq+1]-b[pq+2];
                                        c2t=c[pq+1]-c[pq+2];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;
```

```
                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq+2]-a[pq+3];
                                        b3t=b[pq+2]-b[pq+3];
                                        c3t=c[pq+2]-c[pq+3];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt11=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));
```

```
                                        Fxt1=(-
a2t*(b2t*b3t+c2t*c3t)+a3t*(b2t*b2t+c2t*c2t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b2t*(a2t*a3t+c2t*c3t)+b3t*(a2t*a2t+c2t*c2t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c2t*(b2t*b3t+a2t*a3t)+c3t*(b2t*b2t+a2t*a2t))/(modr1r2C*modr2r3C);

                                        Fxt2=-0.5*cos(phi)*((-
2.0*a1t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr1r2C
));
                                        Fyt2=-0.5*cos(phi)*((-
2.0*b1t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr1r2C
));
                                        Fzt2=-0.5*cos(phi)*((-
2.0*c1t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr1r2C
));


                                        Fxt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                                a1t=a[pq-1]-a[pq];
/*2nd loop*/
                                        b1t=b[pq-1]-b[pq];
                                        c1t=c[pq-1]-c[pq];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq]-a[pq+1];
                                        b2t=b[pq]-b[pq+1];
                                        c2t=c[pq]-c[pq+1];
```

```
                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq+1]-a[pq+2];
                                        b3t=b[pq+1]-b[pq+2];
                                        c3t=c[pq+1]-c[pq+2];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;
```

```
modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));
                                        Vt22=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));
                                        Fxt1=(-
a1t*(b2t*b3t+c2t*c3t)+a2t*(b2t*b3t+c2t*c3t)-a3t*(b1t*b2t+c1t*c2t)-
a3t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b1t*b3t+c1t*c3t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b1t*(a2t*a3t+c2t*c3t)+b2t*(a2t*a3t+c2t*c3t)-b3t*(a1t*a2t+c1t*c2t)-
b3t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a1t*a3t+c1t*c3t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c1t*(b2t*b3t+a2t*a3t)+c2t*(b2t*b3t+a2t*a3t)-c3t*(b1t*b2t+a1t*a2t)-
c3t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b1t*b3t+a1t*a3t))/(modr1r2C*modr2r3C);
Fxt2=-
0.5*cos(phi)*(((2.0*a1t*(b2t*b2t+c2t*c2t)+2.0*a1t*(b1t*b2t+c1t*c2t)-
2.0*a2t*(b1t*b1t+c1t*c1t)-
2.0*a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr1r2C))+((2.0*a3t*(b2t*b3t+c2t*
c3t)-2.0*a2t*(b3t*b3t+c3t*c3t))/(modr2r3C*modr2r3C)));
Fyt2=-
0.5*cos(phi)*(((2.0*b1t*(a2t*a2t+c2t*c2t)+2.0*b1t*(a1t*a2t+c1t*c2t)-
2.0*b2t*(a1t*a1t+c1t*c1t)-
2.0*b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr1r2C))+((2.0*b3t*(a2t*a3t+c2t*
c3t)-2.0*b2t*(a3t*a3t+c3t*c3t))/(modr2r3C*modr2r3C)));
Fzt2=-
0.5*cos(phi)*(((2.0*c1t*(b2t*b2t+a2t*a2t)+2.0*c1t*(b1t*b2t+a1t*a2t)-
2.0*c2t*(b1t*b1t+a1t*a1t)-
2.0*c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr1r2C))+((2.0*c3t*(b2t*b3t+a2t*
a3t)-2.0*c2t*(b3t*b3t+a3t*a3t))/(modr2r3C*modr2r3C)));


                                        Fxt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);


                                        a1t=a[pq-2]-a[pq-1];
                        /*3rd loop*/
                                        b1t=b[pq-2]-b[pq-1];
                                        c1t=c[pq-2]-c[pq-1];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;
```

```
                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq-1]-a[pq];
                                        b2t=b[pq-1]-b[pq];
                                        c2t=c[pq-1]-c[pq];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq]-a[pq+1];
                                        b3t=b[pq]-b[pq+1];
                                        c3t=c[pq]-c[pq+1];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;
```

```
                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                Vt33=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));


Fxt1=(a1t*(b2t*b2t+c2t*c2t)+a1t*(b2t*b3t+c2t*c3t)-
a2t*(b1t*b2t+c1t*c2t)+a3t*(b1t*b2t+c1t*c2t)-
2.0*a2t*(b1t*b3t+c1t*c3t))/(modr1r2C*modr2r3C);

Fyt1=(b1t*(a2t*a2t+c2t*c2t)+b1t*(a2t*a3t+c2t*c3t)-
b2t*(a1t*a2t+c1t*c2t)+b3t*(a1t*a2t+c1t*c2t)-
2.0*b2t*(a1t*a3t+c1t*c3t))/(modr1r2C*modr2r3C);

Fzt1=(c1t*(b2t*b2t+a2t*a2t)+c1t*(b2t*b3t+a2t*a3t)-
c2t*(b1t*b2t+a1t*a2t)+c3t*(b1t*b2t+a1t*a2t)-
2.0*c2t*(b1t*b3t+a1t*a3t))/(modr1r2C*modr2r3C);

Fxt2=-0.5*cos(phi)*(((-
2.0*a1t*(b1t*b2t+c1t*c2t)+2.0*a2t*(b1t*b1t+c1t*c1t))/(modr1r2C*modr1r2C
))+((-2.0*a3t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b3t*b3t+c3t*c3t)-
2.0*a3t*(b2t*b3t+c2t*c3t)+2.0*a2t*(b2t*b3t+c2t*c3t))/(modr2r3C*modr2r3C
)));

Fyt2=-0.5*cos(phi)*(((-
2.0*b1t*(a1t*a2t+c1t*c2t)+2.0*b2t*(a1t*a1t+c1t*c1t))/(modr1r2C*modr1r2C
))+((-2.0*b3t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a3t*a3t+c3t*c3t)-
2.0*b3t*(a2t*a3t+c2t*c3t)+2.0*b2t*(a2t*a3t+c2t*c3t))/(modr2r3C*modr2r3C
)));
```

```
Fzt2=-0.5*cos(phi)*(((-
2.0*c1t*(b1t*b2t+a1t*a2t)+2.0*c2t*(b1t*b1t+a1t*a1t))/(modr1r2C*modr1r2C
))+((-2.0*c3t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b3t*b3t+a3t*a3t)-
2.0*c3t*(b2t*b3t+a2t*a3t)+2.0*c2t*(b2t*b3t+a2t*a3t))/(modr2r3C*modr2r3C
)));


                                        Fxt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                                Vt=Vt11+Vt22+Vt33;

                                        Fxt=Fxt11+Fxt22+Fxt33;
                                        Fyt=Fyt11+Fyt22+Fyt33;
                                        Fzt=Fzt11+Fzt22+Fzt33;

                                  }
                        else
                        if(pq==(nk-1)||lange[pq+1]!=lange[pq])
                        {
                              a1t=a[pq-3]-a[pq-2];
/*4th loop*/
                                        b1t=b[pq-3]-b[pq-2];
                                        c1t=c[pq-3]-c[pq-2];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq-2]-a[pq-1];
                                        b2t=b[pq-2]-b[pq-1];
                                        c2t=c[pq-2]-c[pq-1];
```

```
                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq-1]-a[pq];
                                        b3t=b[pq-1]-b[pq];
                                        c3t=c[pq-1]-c[pq];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;
```

```
modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt44=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));

                                        Fxt1=(-
a1t*(b2t*b2t+c2t*c2t)+a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b1t*(a2t*a2t+c2t*c2t)+b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c1t*(b2t*b2t+a2t*a2t)+c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr2r3C);

              Fxt2=-0.5*cos(phi)*((2.0*a3t*(b2t*b2t+c2t*c2t)-
2.0*a2t*(b2t*b3t+c2t*c3t))/(modr2r3C*modr2r3C));
              Fyt2=-0.5*cos(phi)*((2.0*b3t*(a2t*a2t+c2t*c2t)-
2.0*b2t*(a2t*a3t+c2t*c3t))/(modr2r3C*modr2r3C));
              Fzt2=-0.5*cos(phi)*((2.0*c3t*(b2t*b2t+a2t*a2t)-
2.0*c2t*(b2t*b3t+a2t*a3t))/(modr2r3C*modr2r3C));


                                        Fxt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                              Vt=Vt44;

                                        Fxt=Fxt44;
                                        Fyt=Fyt44;
                                        Fzt=Fzt44;

                        }
                        else
                                if(pq==(nk-
2)||(lange[pq+2]!=lange[pq]&&lange[pq+1]==lange[pq]))
                                {
                              a1t=a[pq-2]-a[pq-1];
/*3rd loop*/
                                        b1t=b[pq-2]-b[pq-1];
                                        c1t=c[pq-2]-c[pq-1];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;
```

```
                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq-1]-a[pq];
                                        b2t=b[pq-1]-b[pq];
                                        c2t=c[pq-1]-c[pq];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq]-a[pq+1];
                                        b3t=b[pq]-b[pq+1];
                                        c3t=c[pq]-c[pq+1];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;
```

```
                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt33=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));


Fxt1=(a1t*(b2t*b2t+c2t*c2t)+a1t*(b2t*b3t+c2t*c3t)-
a2t*(b1t*b2t+c1t*c2t)+a3t*(b1t*b2t+c1t*c2t)-
2.0*a2t*(b1t*b3t+c1t*c3t))/(modr1r2C*modr2r3C);

Fyt1=(b1t*(a2t*a2t+c2t*c2t)+b1t*(a2t*a3t+c2t*c3t)-
b2t*(a1t*a2t+c1t*c2t)+b3t*(a1t*a2t+c1t*c2t)-
2.0*b2t*(a1t*a3t+c1t*c3t))/(modr1r2C*modr2r3C);

Fzt1=(c1t*(b2t*b2t+a2t*a2t)+c1t*(b2t*b3t+a2t*a3t)-
c2t*(b1t*b2t+a1t*a2t)+c3t*(b1t*b2t+a1t*a2t)-
2.0*c2t*(b1t*b3t+a1t*a3t))/(modr1r2C*modr2r3C);

Fxt2=-0.5*cos(phi)*(((-
2.0*a1t*(b1t*b2t+c1t*c2t)+2.0*a2t*(b1t*b1t+c1t*c1t))/(modr1r2C*modr1r2C
))+((-2.0*a3t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b3t*b3t+c3t*c3t)-
2.0*a3t*(b2t*b3t+c2t*c3t)+2.0*a2t*(b2t*b3t+c2t*c3t))/(modr2r3C*modr2r3C
)));

Fyt2=-0.5*cos(phi)*(((-
2.0*b1t*(a1t*a2t+c1t*c2t)+2.0*b2t*(a1t*a1t+c1t*c1t))/(modr1r2C*modr1r2C
))+((-2.0*b3t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a3t*a3t+c3t*c3t)-
2.0*b3t*(a2t*a3t+c2t*c3t)+2.0*b2t*(a2t*a3t+c2t*c3t))/(modr2r3C*modr2r3C
)));
```

```
Fzt2=-0.5*cos(phi)*(((-
2.0*c1t*(b1t*b2t+a1t*a2t)+2.0*c2t*(b1t*b1t+a1t*a1t))/(modr1r2C*modr1r2C
))+((-2.0*c3t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b3t*b3t+a3t*a3t)-
2.0*c3t*(b2t*b3t+a2t*a3t)+2.0*c2t*(b2t*b3t+a2t*a3t))/(modr2r3C*modr2r3C
)));


                                        Fxt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);


                                a1t=a[pq-3]-a[pq-2];
/*4th loop*/
                                        b1t=b[pq-3]-b[pq-2];
                                        c1t=c[pq-3]-c[pq-2];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq-2]-a[pq-1];
                                        b2t=b[pq-2]-b[pq-1];
                                        c2t=c[pq-2]-c[pq-1];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
```

```
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq-1]-a[pq];
                                        b3t=b[pq-1]-b[pq];
                                        c3t=c[pq-1]-c[pq];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));
```

```
                                        Vt44=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));

                                        Fxt1=(-
a1t*(b2t*b2t+c2t*c2t)+a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b1t*(a2t*a2t+c2t*c2t)+b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c1t*(b2t*b2t+a2t*a2t)+c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr2r3C);

              Fxt2=-0.5*cos(phi)*((2.0*a3t*(b2t*b2t+c2t*c2t)-
2.0*a2t*(b2t*b3t+c2t*c3t))/(modr2r3C*modr2r3C));
              Fyt2=-0.5*cos(phi)*((2.0*b3t*(a2t*a2t+c2t*c2t)-
2.0*b2t*(a2t*a3t+c2t*c3t))/(modr2r3C*modr2r3C));
              Fzt2=-0.5*cos(phi)*((2.0*c3t*(b2t*b2t+a2t*a2t)-
2.0*c2t*(b2t*b3t+a2t*a3t))/(modr2r3C*modr2r3C));

                                        Fxt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                                Vt=Vt33+Vt44;

                                        Fxt=Fxt33+Fxt44;
                                        Fyt=Fyt33+Fyt44;
                                        Fzt=Fzt33+Fzt44;

                        }
                        else
                                if(pq==(nk-
3)||(lange[pq+3]!=lange[pq]&&lange[pq+2]==lange[pq]))
                                {
                                a1t=a[pq-1]-a[pq];
/*2nd loop*/
                                        b1t=b[pq-1]-b[pq];
                                        c1t=c[pq-1]-c[pq];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
```

```
                                        c1t=c1t-boxZ;
                                else
                                if(c1t<-(0.5*boxZ))
                                        c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                a2t=a[pq]-a[pq+1];
                                b2t=b[pq]-b[pq+1];
                                c2t=c[pq]-c[pq+1];

                                if(a2t>(0.5*boxX))
                                        a2t=a2t-boxX;
                                else
                                if(a2t<-(0.5*boxX))
                                        a2t=boxX+a2t;

                                if(b2t>(0.5*boxY))
                                        b2t=b2t-boxY;
                                else
                                if(b2t<-(0.5*boxY))
                                        b2t=boxY+b2t;

                                if(c2t>(0.5*boxZ))
                                        c2t=c2t-boxZ;
                                else
                                if(c2t<-(0.5*boxZ))
                                        c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                        a3t=a[pq+1]-a[pq+2];
                                b3t=b[pq+1]-b[pq+2];
                                c3t=c[pq+1]-c[pq+2];

                                if(a3t>(0.5*boxX))
                                        a3t=a3t-boxX;
                                else
                                if(a3t<-(0.5*boxX))
                                        a3t=boxX+a3t;

                                if(b3t>(0.5*boxY))
                                        b3t=b3t-boxY;
                                else
                                if(b3t<-(0.5*boxY))
                                        b3t=boxY+b3t;

                                if(c3t>(0.5*boxZ))
                                        c3t=c3t-boxZ;
                                else
                                if(c3t<-(0.5*boxZ))
                                        c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);
```

```
                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt22=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));
                                        Fxt1=(-
a1t*(b2t*b3t+c2t*c3t)+a2t*(b2t*b3t+c2t*c3t)-a3t*(b1t*b2t+c1t*c2t)-
a3t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b1t*b3t+c1t*c3t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b1t*(a2t*a3t+c2t*c3t)+b2t*(a2t*a3t+c2t*c3t)-b3t*(a1t*a2t+c1t*c2t)-
b3t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a1t*a3t+c1t*c3t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c1t*(b2t*b3t+a2t*a3t)+c2t*(b2t*b3t+a2t*a3t)-c3t*(b1t*b2t+a1t*a2t)-
c3t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b1t*b3t+a1t*a3t))/(modr1r2C*modr2r3C);

Fxt2=-
0.5*cos(phi)*(((2.0*a1t*(b2t*b2t+c2t*c2t)+2.0*a1t*(b1t*b2t+c1t*c2t)-
2.0*a2t*(b1t*b1t+c1t*c1t)-
2.0*a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr1r2C))+((2.0*a3t*(b2t*b3t+c2t*
c3t)-2.0*a2t*(b3t*b3t+c3t*c3t))/(modr2r3C*modr2r3C)));

Fyt2=-
0.5*cos(phi)*(((2.0*b1t*(a2t*a2t+c2t*c2t)+2.0*b1t*(a1t*a2t+c1t*c2t)-
2.0*b2t*(a1t*a1t+c1t*c1t)-
2.0*b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr1r2C))+((2.0*b3t*(a2t*a3t+c2t*
c3t)-2.0*b2t*(a3t*a3t+c3t*c3t))/(modr2r3C*modr2r3C)));

Fzt2=-
0.5*cos(phi)*(((2.0*c1t*(b2t*b2t+a2t*a2t)+2.0*c1t*(b1t*b2t+a1t*a2t)-
2.0*c2t*(b1t*b1t+a1t*a1t)-
2.0*c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr1r2C))+((2.0*c3t*(b2t*b3t+a2t*
a3t)-2.0*c2t*(b3t*b3t+a3t*a3t))/(modr2r3C*modr2r3C)));


                                        Fxt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
```

```
                                        Fzt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                                a1t=a[pq-2]-a[pq-1];
/*3rd loop*/
                                        b1t=b[pq-2]-b[pq-1];
                                        c1t=c[pq-2]-c[pq-1];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq-1]-a[pq];
                                        b2t=b[pq-1]-b[pq];
                                        c2t=c[pq-1]-c[pq];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq]-a[pq+1];
                                        b3t=b[pq]-b[pq+1];
```

```
                                        c3t=c[pq]-c[pq+1];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt33=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));


Fxt1=(a1t*(b2t*b2t+c2t*c2t)+a1t*(b2t*b3t+c2t*c3t)-
a2t*(b1t*b2t+c1t*c2t)+a3t*(b1t*b2t+c1t*c2t)-
2.0*a2t*(b1t*b3t+c1t*c3t))/(modr1r2C*modr2r3C);

Fyt1=(b1t*(a2t*a2t+c2t*c2t)+b1t*(a2t*a3t+c2t*c3t)-
b2t*(a1t*a2t+c1t*c2t)+b3t*(a1t*a2t+c1t*c2t)-
2.0*b2t*(a1t*a3t+c1t*c3t))/(modr1r2C*modr2r3C);

Fzt1=(c1t*(b2t*b2t+a2t*a2t)+c1t*(b2t*b3t+a2t*a3t)-
c2t*(b1t*b2t+a1t*a2t)+c3t*(b1t*b2t+a1t*a2t)-
2.0*c2t*(b1t*b3t+a1t*a3t))/(modr1r2C*modr2r3C);
```

```
Fxt2=-0.5*cos(phi)*(((-
2.0*a1t*(b1t*b2t+c1t*c2t)+2.0*a2t*(b1t*b1t+c1t*c1t))/(modr1r2C*modr1r2C
))+((-2.0*a3t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b3t*b3t+c3t*c3t)-
2.0*a3t*(b2t*b3t+c2t*c3t)+2.0*a2t*(b2t*b3t+c2t*c3t))/(modr2r3C*modr2r3C
)));

Fyt2=-0.5*cos(phi)*(((-
2.0*b1t*(a1t*a2t+c1t*c2t)+2.0*b2t*(a1t*a1t+c1t*c1t))/(modr1r2C*modr1r2C
))+((-2.0*b3t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a3t*a3t+c3t*c3t)-
2.0*b3t*(a2t*a3t+c2t*c3t)+2.0*b2t*(a2t*a3t+c2t*c3t))/(modr2r3C*modr2r3C
)));

Fzt2=-0.5*cos(phi)*(((-
2.0*c1t*(b1t*b2t+a1t*a2t)+2.0*c2t*(b1t*b1t+a1t*a1t))/(modr1r2C*modr1r2C
))+((-2.0*c3t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b3t*b3t+a3t*a3t)-
2.0*c3t*(b2t*b3t+a2t*a3t)+2.0*c2t*(b2t*b3t+a2t*a3t))/(modr2r3C*modr2r3C
)));


                                        Fxt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);


                                a1t=a[pq-3]-a[pq-2];
/*4th loop*/
                                        b1t=b[pq-3]-b[pq-2];
                                        c1t=c[pq-3]-c[pq-2];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);
```

```
                                        a2t=a[pq-2]-a[pq-1];
                                        b2t=b[pq-2]-b[pq-1];
                                        c2t=c[pq-2]-c[pq-1];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq-1]-a[pq];
                                        b3t=b[pq-1]-b[pq];
                                        c3t=c[pq-1]-c[pq];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);
```

```
                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt44=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));

                                        Fxt1=(-
a1t*(b2t*b2t+c2t*c2t)+a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b1t*(a2t*a2t+c2t*c2t)+b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c1t*(b2t*b2t+a2t*a2t)+c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr2r3C);

              Fxt2=-0.5*cos(phi)*((2.0*a3t*(b2t*b2t+c2t*c2t)-
2.0*a2t*(b2t*b3t+c2t*c3t))/(modr2r3C*modr2r3C));
              Fyt2=-0.5*cos(phi)*((2.0*b3t*(a2t*a2t+c2t*c2t)-
2.0*b2t*(a2t*a3t+c2t*c3t))/(modr2r3C*modr2r3C));
              Fzt2=-0.5*cos(phi)*((2.0*c3t*(b2t*b2t+a2t*a2t)-
2.0*c2t*(b2t*b3t+a2t*a3t))/(modr2r3C*modr2r3C));

                                        Fxt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                              Vt=Vt22+Vt33+Vt44;

                                        Fxt=Fxt22+Fxt33+Fxt44;
                                        Fyt=Fyt22+Fyt33+Fyt44;
                                        Fzt=Fzt22+Fzt33+Fzt44;

                        }
                        else
                              {
                              a1t=a[pq]-a[pq+1];
/*1st loop*/
                                        b1t=b[pq]-b[pq+1];
                                        c1t=c[pq]-c[pq+1];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
```

```
                                        a1t=boxX+a1t;

                                if(b1t>(0.5*boxY))
                                        b1t=b1t-boxY;
                                else
                                if(b1t<-(0.5*boxY))
                                        b1t=boxY+b1t;

                                if(c1t>(0.5*boxZ))
                                        c1t=c1t-boxZ;
                                else
                                if(c1t<-(0.5*boxZ))
                                        c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                a2t=a[pq+1]-a[pq+2];
                                b2t=b[pq+1]-b[pq+2];
                                c2t=c[pq+1]-c[pq+2];

                                if(a2t>(0.5*boxX))
                                        a2t=a2t-boxX;
                                else
                                if(a2t<-(0.5*boxX))
                                        a2t=boxX+a2t;

                                if(b2t>(0.5*boxY))
                                        b2t=b2t-boxY;
                                else
                                if(b2t<-(0.5*boxY))
                                        b2t=boxY+b2t;

                                if(c2t>(0.5*boxZ))
                                        c2t=c2t-boxZ;
                                else
                                if(c2t<-(0.5*boxZ))
                                        c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                        a3t=a[pq+2]-a[pq+3];
                                b3t=b[pq+2]-b[pq+3];
                                c3t=c[pq+2]-c[pq+3];

                                if(a3t>(0.5*boxX))
                                        a3t=a3t-boxX;
                                else
                                if(a3t<-(0.5*boxX))
                                        a3t=boxX+a3t;

                                if(b3t>(0.5*boxY))
                                        b3t=b3t-boxY;
                                else
                                if(b3t<-(0.5*boxY))
                                        b3t=boxY+b3t;
```

```
                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt11=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));


                                        Fxt1=(-
a2t*(b2t*b3t+c2t*c3t)+a3t*(b2t*b2t+c2t*c2t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b2t*(a2t*a3t+c2t*c3t)+b3t*(a2t*a2t+c2t*c2t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c2t*(b2t*b3t+a2t*a3t)+c3t*(b2t*b2t+a2t*a2t))/(modr1r2C*modr2r3C);

                                        Fxt2=-0.5*cos(phi)*((-
2.0*a1t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr1r2C
));
                                        Fyt2=-0.5*cos(phi)*((-
2.0*b1t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr1r2C
));
                                        Fzt2=-0.5*cos(phi)*((-
2.0*c1t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr1r2C
));


                                        Fxt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
```

```
                                        Fzt11=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);


                                a1t=a[pq-1]-a[pq];
/*2nd loop*/
                                        b1t=b[pq-1]-b[pq];
                                        c1t=c[pq-1]-c[pq];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq]-a[pq+1];
                                        b2t=b[pq]-b[pq+1];
                                        c2t=c[pq]-c[pq+1];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq+1]-a[pq+2];
```

```
                                        b3t=b[pq+1]-b[pq+2];
                                        c3t=c[pq+1]-c[pq+2];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt22=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));
                                        Fxt1=(-
a1t*(b2t*b3t+c2t*c3t)+a2t*(b2t*b3t+c2t*c3t)-a3t*(b1t*b2t+c1t*c2t)-
a3t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b1t*b3t+c1t*c3t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b1t*(a2t*a3t+c2t*c3t)+b2t*(a2t*a3t+c2t*c3t)-b3t*(a1t*a2t+c1t*c2t)-
b3t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a1t*a3t+c1t*c3t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c1t*(b2t*b3t+a2t*a3t)+c2t*(b2t*b3t+a2t*a3t)-c3t*(b1t*b2t+a1t*a2t)-
c3t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b1t*b3t+a1t*a3t))/(modr1r2C*modr2r3C);

Fxt2=-
0.5*cos(phi)*(((2.0*a1t*(b2t*b2t+c2t*c2t)+2.0*a1t*(b1t*b2t+c1t*c2t)-
```

```
2.0*a2t*(b1t*b1t+c1t*c1t)-
2.0*a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr1r2C))+((2.0*a3t*(b2t*b3t+c2t*
c3t)-2.0*a2t*(b3t*b3t+c3t*c3t))/(modr2r3C*modr2r3C)));

Fyt2=-
0.5*cos(phi)*(((2.0*b1t*(a2t*a2t+c2t*c2t)+2.0*b1t*(a1t*a2t+c1t*c2t)-
2.0*b2t*(a1t*a1t+c1t*c1t)-
2.0*b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr1r2C))+((2.0*b3t*(a2t*a3t+c2t*
c3t)-2.0*b2t*(a3t*a3t+c3t*c3t))/(modr2r3C*modr2r3C)));

Fzt2=-
0.5*cos(phi)*(((2.0*c1t*(b2t*b2t+a2t*a2t)+2.0*c1t*(b1t*b2t+a1t*a2t)-
2.0*c2t*(b1t*b1t+a1t*a1t)-
2.0*c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr1r2C))+((2.0*c3t*(b2t*b3t+a2t*
a3t)-2.0*c2t*(b3t*b3t+a3t*a3t))/(modr2r3C*modr2r3C)));


                                        Fxt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt22=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);


                                a1t=a[pq-2]-a[pq-1];
/*3rd loop*/
                                        b1t=b[pq-2]-b[pq-1];
                                        c1t=c[pq-2]-c[pq-1];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq-1]-a[pq];
                                        b2t=b[pq-1]-b[pq];
                                        c2t=c[pq-1]-c[pq];
```

```
                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq]-a[pq+1];
                                        b3t=b[pq]-b[pq+1];
                                        c3t=c[pq]-c[pq+1];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                        r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;
```

```
modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                        Vt33=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));


Fxt1=(a1t*(b2t*b2t+c2t*c2t)+a1t*(b2t*b3t+c2t*c3t)-
a2t*(b1t*b2t+c1t*c2t)+a3t*(b1t*b2t+c1t*c2t)-
2.0*a2t*(b1t*b3t+c1t*c3t))/(modr1r2C*modr2r3C);

Fyt1=(b1t*(a2t*a2t+c2t*c2t)+b1t*(a2t*a3t+c2t*c3t)-
b2t*(a1t*a2t+c1t*c2t)+b3t*(a1t*a2t+c1t*c2t)-
2.0*b2t*(a1t*a3t+c1t*c3t))/(modr1r2C*modr2r3C);

Fzt1=(c1t*(b2t*b2t+a2t*a2t)+c1t*(b2t*b3t+a2t*a3t)-
c2t*(b1t*b2t+a1t*a2t)+c3t*(b1t*b2t+a1t*a2t)-
2.0*c2t*(b1t*b3t+a1t*a3t))/(modr1r2C*modr2r3C);

Fxt2=-0.5*cos(phi)*(((-
2.0*a1t*(b1t*b2t+c1t*c2t)+2.0*a2t*(b1t*b1t+c1t*c1t))/(modr1r2C*modr1r2C
))+((-2.0*a3t*(b2t*b2t+c2t*c2t)+2.0*a2t*(b3t*b3t+c3t*c3t)-
2.0*a3t*(b2t*b3t+c2t*c3t)+2.0*a2t*(b2t*b3t+c2t*c3t))/(modr2r3C*modr2r3C
)));

Fyt2=-0.5*cos(phi)*(((-
2.0*b1t*(a1t*a2t+c1t*c2t)+2.0*b2t*(a1t*a1t+c1t*c1t))/(modr1r2C*modr1r2C
))+((-2.0*b3t*(a2t*a2t+c2t*c2t)+2.0*b2t*(a3t*a3t+c3t*c3t)-
2.0*b3t*(a2t*a3t+c2t*c3t)+2.0*b2t*(a2t*a3t+c2t*c3t))/(modr2r3C*modr2r3C
)));

Fzt2=-0.5*cos(phi)*(((-
2.0*c1t*(b1t*b2t+a1t*a2t)+2.0*c2t*(b1t*b1t+a1t*a1t))/(modr1r2C*modr1r2C
))+((-2.0*c3t*(b2t*b2t+a2t*a2t)+2.0*c2t*(b3t*b3t+a3t*a3t)-
2.0*c3t*(b2t*b3t+a2t*a3t)+2.0*c2t*(b2t*b3t+a2t*a3t))/(modr2r3C*modr2r3C
)));


                                        Fxt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt33=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);
```

```
                                        a1t=a[pq-3]-a[pq-2];
                                /*4th loop*/
                                        b1t=b[pq-3]-b[pq-2];
                                        c1t=c[pq-3]-c[pq-2];

                                        if(a1t>(0.5*boxX))
                                                a1t=a1t-boxX;
                                        else
                                        if(a1t<-(0.5*boxX))
                                                a1t=boxX+a1t;

                                        if(b1t>(0.5*boxY))
                                                b1t=b1t-boxY;
                                        else
                                        if(b1t<-(0.5*boxY))
                                                b1t=boxY+b1t;

                                        if(c1t>(0.5*boxZ))
                                                c1t=c1t-boxZ;
                                        else
                                        if(c1t<-(0.5*boxZ))
                                                c1t=boxZ+c1t;


r1t=pow(((a1t*a1t)+(b1t*b1t)+(c1t*c1t)),0.5);

                                        a2t=a[pq-2]-a[pq-1];
                                        b2t=b[pq-2]-b[pq-1];
                                        c2t=c[pq-2]-c[pq-1];

                                        if(a2t>(0.5*boxX))
                                                a2t=a2t-boxX;
                                        else
                                        if(a2t<-(0.5*boxX))
                                                a2t=boxX+a2t;

                                        if(b2t>(0.5*boxY))
                                                b2t=b2t-boxY;
                                        else
                                        if(b2t<-(0.5*boxY))
                                                b2t=boxY+b2t;

                                        if(c2t>(0.5*boxZ))
                                                c2t=c2t-boxZ;
                                        else
                                        if(c2t<-(0.5*boxZ))
                                                c2t=boxZ+c2t;


r2t=pow(((a2t*a2t)+(b2t*b2t)+(c2t*c2t)),0.5);

                                a3t=a[pq-1]-a[pq];
                                        b3t=b[pq-1]-b[pq];
                                        c3t=c[pq-1]-c[pq];

                                        if(a3t>(0.5*boxX))
                                                a3t=a3t-boxX;
```

```
                                        else
                                        if(a3t<-(0.5*boxX))
                                                a3t=boxX+a3t;

                                        if(b3t>(0.5*boxY))
                                                b3t=b3t-boxY;
                                        else
                                        if(b3t<-(0.5*boxY))
                                                b3t=boxY+b3t;

                                        if(c3t>(0.5*boxZ))
                                                c3t=c3t-boxZ;
                                        else
                                        if(c3t<-(0.5*boxZ))
                                                c3t=boxZ+c3t;


r3t=pow(((a3t*a3t)+(b3t*b3t)+(c3t*c3t)),0.5);

                                r1r2Cx=b1t*c2t-b2t*c1t;
                                        r1r2Cy=-(a1t*c2t-a2t*c1t);
                                        r1r2Cz=a1t*b2t-a2t*b1t;


modr1r2C=pow((r1r2Cx*r1r2Cx+r1r2Cy*r1r2Cy+r1r2Cz*r1r2Cz),0.5);

                                        r2r3Cx=b2t*c3t-b3t*c2t;
                                        r2r3Cy=-(a2t*c3t-a3t*c2t);
                                        r2r3Cz=a2t*b3t-a3t*b2t;


modr2r3C=pow((r2r3Cx*r2r3Cx+r2r3Cy*r2r3Cy+r2r3Cz*r2r3Cz),0.5);


phi=acos((r1r2Cx*r2r3Cx+r1r2Cy*r2r3Cy+r1r2Cz*r2r3Cz)/(modr1r2C*modr2r3C
));

                                Vt44=0.1*(rt01-rt02*cos(phi)-
rt03*cos(phi)*cos(phi)+rt04*pow(cos(phi),3.0)+rt05*pow(cos(phi),4.0)+rt
06*pow(cos(phi),5.0));

                                        Fxt1=(-
a1t*(b2t*b2t+c2t*c2t)+a2t*(b1t*b2t+c1t*c2t))/(modr1r2C*modr2r3C);
                                        Fyt1=(-
b1t*(a2t*a2t+c2t*c2t)+b2t*(a1t*a2t+c1t*c2t))/(modr1r2C*modr2r3C);
                                        Fzt1=(-
c1t*(b2t*b2t+a2t*a2t)+c2t*(b1t*b2t+a1t*a2t))/(modr1r2C*modr2r3C);

              Fxt2=-0.5*cos(phi)*((2.0*a3t*(b2t*b2t+c2t*c2t)-
2.0*a2t*(b2t*b3t+c2t*c3t))/(modr2r3C*modr2r3C));
              Fyt2=-0.5*cos(phi)*((2.0*b3t*(a2t*a2t+c2t*c2t)-
2.0*b2t*(a2t*a3t+c2t*c3t))/(modr2r3C*modr2r3C));
              Fzt2=-0.5*cos(phi)*((2.0*c3t*(b2t*b2t+a2t*a2t)-
2.0*c2t*(b2t*b3t+a2t*a3t))/(modr2r3C*modr2r3C));
```

```
                                        Fxt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fxt1+Fxt2);
                                        Fyt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fyt1+Fyt2);
                                        Fzt44=(-rt02-
2.0*rt03*cos(phi)+3.0*rt04*cos(phi)*cos(phi)+4.0*rt05*pow(cos(phi),3.0)
+5.0*rt06*pow(cos(phi),4.0))*(Fzt1+Fzt2);

                              Vt=Vt11+Vt22+Vt33+Vt44;

                              Fxt=Fxt11+Fxt22+Fxt33+Fxt44;
                              Fyt=Fyt11+Fyt22+Fyt33+Fyt44;
                              Fzt=Fzt11+Fzt22+Fzt33+Fzt44;

                                }

                         /*Streching*/
                                    if(pq!=0)
                                    {
                                          ar1=a[pq]-a[pq-1];
                                          br1=b[pq]-b[pq-1];
                                          cr1=c[pq]-c[pq-1];

                                          if(ar1>(0.5*boxX))
                                                ar1=ar1-boxX;
                                          else
                                                if(ar1<-(0.5*boxX))
                                                      ar1=boxX+ar1;

                                          if(br1>(0.5*boxY))
                                                br1=br1-boxY;
                                          else
                                                if(br1<-(0.5*boxY))
                                                      br1=boxY+br1;

                                          if(cr1>(0.5*boxZ))
                                                cr1=cr1-boxZ;
                                          else
                                                if(cr1<-(0.5*boxZ))
                                                      cr1=boxZ+cr1;

      rs1=pow(((ar1*ar1)+(br1*br1)+(cr1*cr1)),0.5);
                                    }

                                    if(pq!=(nk-1))
                                    {
                                          ar2=a[pq]-a[pq+1];
                                          br2=b[pq]-b[pq+1];
                                          cr2=c[pq]-c[pq+1];

                                          if(ar2>(0.5*boxX))
                                                ar2=ar2-boxX;
                                          else
                                                if(ar2<-(0.5*boxX))
                                                      ar2=boxX+ar2;
```

```
                                        if(br2>(0.5*boxY))
                                              br2=br2-boxY;
                                        else
                                              if(br2<-(0.5*boxY))
                                                    br2=boxY+br2;

                                        if(cr2>(0.5*boxZ))
                                              cr2=cr2-boxZ;
                                        else
                                              if(cr2<-(0.5*boxZ))
                                                    cr2=boxZ+cr2;

      rs2=pow(((ar2*ar2)+(br2*br2)+(cr2*cr2)),0.5);
                                  }

                                  if((pq==0)||(lange[pq-
1]!=lange[pq]))
                                  {
                                        Vs=0.5*rbond*pow((b0-
rs2),2.0);

                                        Fxs=rbond*ar2*((b0/rs2)-1.0);
                                        Fys=rbond*br2*((b0/rs2)-1.0);
                                        Fzs=rbond*cr2*((b0/rs2)-1.0);
                                  }
                                  else
                                  if((pq==(nk-
1))||(lange[pq+1]!=lange[pq]))
                                  {
                                        Vs=0.5*rbond*pow((b0-
rs1),2.0);

                                        Fxs=rbond*ar1*((b0/rs1)-1.0);
                                        Fys=rbond*br1*((b0/rs1)-1.0);
                                        Fzs=rbond*cr1*((b0/rs1)-1.0);
                                  }
                                  else
                                  {
                                        Vs=0.5*rbond*pow((2.0*b0-rs1-
rs2),2.0);

                                        Fxs=rbond*ar1*((b0/rs1)-
1.0)+rbond*ar2*((b0/rs2)-1.0);
                                        Fys=rbond*br1*((b0/rs1)-
1.0)+rbond*br2*((b0/rs2)-1.0);
                                        Fzs=rbond*cr1*((b0/rs1)-
1.0)+rbond*cr2*((b0/rs2)-1.0);
                                  }

                  }
                        /*Water molecule-Intramolecular interactions-
Toukan, K., et al. (1985). Physical Review B 31: 2643-2648*/
                  if(sele[pq]==3)
                  {
                        a1x=a[pq]-a[pq+1];
                        b1y=b[pq]-b[pq+1];
```

```
c1z=c[pq]-c[pq+1];

if(a1x>(0.5*boxX))
      a1x=a1x-boxX;
else
      if(a1x<-(0.5*boxX))
            a1x=boxX+a1x;

if(b1y>(0.5*boxY))
      b1y=b1y-boxY;
else
      if(b1y<-(0.5*boxY))
            b1y=boxY+b1y;

if(c1z>(0.5*boxZ))
      c1z=c1z-boxZ;
else
      if(c1z<-(0.5*boxZ))
            c1z=boxZ+c1z;

r1=pow(((a1x*a1x)+(b1y*b1y)+(c1z*c1z)),0.5);

a2x=a[pq]-a[pq+2];
b2y=b[pq]-b[pq+2];
c2z=c[pq]-c[pq+2];

if(a2x>(0.5*boxX))
      a2x=a2x-boxX;
else
      if(a2x<-(0.5*boxX))
            a2x=boxX+a2x;

if(b2y>(0.5*boxY))
      b2y=b2y-boxY;
else
      if(b2y<-(0.5*boxY))
            b2y=boxY+b2y;

if(c2z>(0.5*boxZ))
      c2z=c2z-boxZ;
else
      if(c2z<-(0.5*boxZ))
            c2z=boxZ+c2z;

r2=pow(((a2x*a2x)+(b2y*b2y)+(c2z*c2z)),0.5);

a3x=a[pq+1]-a[pq+2];
b3y=b[pq+1]-b[pq+2];
c3z=c[pq+1]-c[pq+2];

if(a3x>(0.5*boxX))
      a3x=a3x-boxX;
else
      if(a3x<-(0.5*boxX))
            a3x=boxX+a3x;

if(b3y>(0.5*boxY))
```

```
                        b3y=b3y-boxY;
                  else
                        if(b3y<-(0.5*boxY))
                              b3y=boxY+b3y;

                  if(c3z>(0.5*boxZ))
                        c3z=c3z-boxZ;
                  else
                        if(c3z<-(0.5*boxZ))
                              c3z=boxZ+c3z;
                  r3=pow(((a3x*a3x)+(b3y*b3y)+(c3z*c3z)),0.5);

                  ex1=-2.566*(r1-1.0);

                  /*
Voh1=4.4195*pow((1-exp(ex1)),2.0);
                  Fx1=(-2.0*2.361*4.4195*(1-
exp(ex1))*(exp(ex1))*(a1x/r1));
                  Fy1=(-2.0*2.361*4.4195*(1-
exp(ex1))*(exp(ex1))*(b1y/r1));
                  Fz1=(-2.0*2.361*4.4195*(1-
exp(ex1))*(exp(ex1))*(c1z/r1));

                  ex2=-2.566*(r2-1.0);

                  Voh2=4.4195*pow((1-exp(ex2)),2.0);
                  Fx2=(-2.0*2.361*4.4195*(1-
exp(ex2))*(exp(ex2))*(a2x/r2));
                  Fy2=(-2.0*2.361*4.4195*(1-
exp(ex2))*(exp(ex2))*(b2y/r2));
                  Fz2=(-2.0*2.361*4.4195*(1-
exp(ex2))*(exp(ex2))*(c2z/r2));
                  */

                  Voh1=Rw*Rw*Dw*pow((r1-1.0),2.0)+0.5*bq*pow((r3-
1.6),2.0)+cq*(r1+r2-2.0)*(r3-1.6)+dq*(r1-1.0)*(r2-1.0);
                  Fx1=-(a1x/r1)*(2.0*Rw*Rw*Dw*(r1-1.0)+cq*(r3-
1.6)+dq*(r2-1.0));
                  Fy1=-(b1y/r1)*(2.0*Rw*Rw*Dw*(r1-1.0)+cq*(r3-
1.6)+dq*(r2-1.0));
                  Fz1=-(c1z/r1)*(2.0*Rw*Rw*Dw*(r1-1.0)+cq*(r3-
1.6)+dq*(r2-1.0));

                  Voh2=Rw*Rw*Dw*pow((r2-1.0),2.0)+0.5*bq*pow((r3-
1.6),2.0)+cq*(r1+r2-2.0)*(r3-1.6)+dq*(r1-1.0)*(r2-1.0);
                  Fx2=-(a2x/r2)*(2.0*Rw*Rw*Dw*(r2-1.0)+cq*(r3-
1.6)+dq*(r1-1.0));
                  Fy2=-(b2y/r2)*(2.0*Rw*Rw*Dw*(r2-1.0)+cq*(r3-
1.6)+dq*(r1-1.0));
                  Fz2=-(c2z/r2)*(2.0*Rw*Rw*Dw*(r2-1.0)+cq*(r3-
1.6)+dq*(r1-1.0));

                  Vo=Voh1+Voh2;
                  Fxw=Fx1+Fx2;
                  Fyw=Fy1+Fy2;
                  Fzw=Fz1+Fz2;
```

```
}

if(sele[pq]==2)
{
      a1x=a[pq]-a[pq-1];
      b1y=b[pq]-b[pq-1];
      c1z=c[pq]-c[pq-1];

      if(a1x>(0.5*boxX))
            a1x=a1x-boxX;
      else
            if(a1x<-(0.5*boxX))
                  a1x=boxX+a1x;

      if(b1y>(0.5*boxY))
            b1y=b1y-boxY;
      else
            if(b1y<-(0.5*boxY))
                  b1y=boxY+b1y;

      if(c1z>(0.5*boxZ))
            c1z=c1z-boxZ;
      else
            if(c1z<-(0.5*boxZ))
                  c1z=boxZ+c1z;

      r1=pow(((a1x*a1x)+(b1y*b1y)+(c1z*c1z)),0.5);

      a2x=a[pq-1]-a[pq+1];
      b2y=b[pq-1]-b[pq+1];
      c2z=c[pq-1]-c[pq+1];

      if(a2x>(0.5*boxX))
            a2x=a2x-boxX;
      else
            if(a2x<-(0.5*boxX))
                  a2x=boxX+a2x;

      if(b2y>(0.5*boxY))
            b2y=b2y-boxY;
      else
            if(b2y<-(0.5*boxY))
                  b2y=boxY+b2y;

      if(c2z>(0.5*boxZ))
            c2z=c2z-boxZ;
      else
            if(c2z<-(0.5*boxZ))
                  c2z=boxZ+c2z;

      r2=pow(((a2x*a2x)+(b2y*b2y)+(c2z*c2z)),0.5);

      a3x=a[pq]-a[pq+1];
      b3y=b[pq]-b[pq+1];
      c3z=c[pq]-c[pq+1];

      if(a3x>(0.5*boxX))
```

```
                        a3x=a3x-boxX;
                  else
                        if(a3x<-(0.5*boxX))
                              a3x=boxX+a3x;

                  if(b3y>(0.5*boxY))
                        b3y=b3y-boxY;
                  else
                        if(b3y<-(0.5*boxY))
                              b3y=boxY+b3y;

                  if(c3z>(0.5*boxZ))
                        c3z=c3z-boxZ;
                  else
                        if(c3z<-(0.5*boxZ))
                              c3z=boxZ+c3z;
                  r3=pow(((a3x*a3x)+(b3y*b3y)+(c3z*c3z)),0.5);

                  ex1=-2.566*(r1-1.0);

                  /*
Voh1=4.4195*pow((1-exp(ex1)),2.0);
                  Fx1=(-2.0*2.361*4.4195*(1-
exp(ex1))*(exp(ex1))*(a1x/r1));
                  Fy1=(-2.0*2.361*4.4195*(1-
exp(ex1))*(exp(ex1))*(b1y/r1));
                  Fz1=(-2.0*2.361*4.4195*(1-
exp(ex1))*(exp(ex1))*(c1z/r1));


      Vhh=0.5*aq*((r1*r1)+(r2*r2))+0.5*bq*(r3*r3)+cq*(r1+r2)*r3+dq*r1*r
2;
                  Fx2=-(a3x/r3)*(bq*r3+cq*r1+cq*r2);
                  Fy2=-(b3y/r3)*(bq*r3+cq*r1+cq*r2);
                  Fz2=-(c3z/r3)*(bq*r3+cq*r1+cq*r2);
                  */

                  Voh1=Rw*Rw*Dw*pow((r1-1.0),2.0)+0.5*bq*pow((r3-
1.6),2.0)+cq*(r1+r2-2.0)*(r3-1.6)+dq*(r1-1.0)*(r2-1.0);
                  Fx1=-(a1x/r1)*(2.0*Rw*Rw*Dw*(r1-1.0)+cq*(r3-
1.6)+dq*(r2-1.0));
                  Fy1=-(b1y/r1)*(2.0*Rw*Rw*Dw*(r1-1.0)+cq*(r3-
1.6)+dq*(r2-1.0));
                  Fz1=-(c1z/r1)*(2.0*Rw*Rw*Dw*(r1-1.0)+cq*(r3-
1.6)+dq*(r2-1.0));

                  Vhh=Rw*Rw*Dw*pow((r1-1.0),2.0)+0.5*bq*pow((r3-
1.6),2.0)+cq*(r1+r2-2.0)*(r3-1.6)+dq*(r1-1.0)*(r2-1.0);
                  Fx2=-(a3x/r3)*(bq*(r3-1.6)+cq*(r1-1.0)+cq*(r2-
1.0));
                  Fy2=-(b3y/r3)*(bq*(r3-1.6)+cq*(r1-1.0)+cq*(r2-
1.0));
                  Fz2=-(c3z/r3)*(bq*(r3-1.6)+cq*(r1-1.0)+cq*(r2-
1.0));

                  Vh=Vhh+Voh1;
                  Fxw=Fx1+Fx2;
```

```
		Fyw=Fy1+Fy2;
		Fzw=Fz1+Fz2;
	}
	if(sele[pq]==5)
	{
		a1x=a[pq-2]-a[pq-1];
		b1y=b[pq-2]-b[pq-1];
		c1z=c[pq-2]-c[pq-1];

		if(a1x>(0.5*boxX))
			a1x=a1x-boxX;
		else
			if(a1x<-(0.5*boxX))
				a1x=boxX+a1x;

		if(b1y>(0.5*boxY))
			b1y=b1y-boxY;
		else
			if(b1y<-(0.5*boxY))
				b1y=boxY+b1y;

		if(c1z>(0.5*boxZ))
			c1z=c1z-boxZ;
		else
			if(c1z<-(0.5*boxZ))
				c1z=boxZ+c1z;
		r1=pow(((a1x*a1x)+(b1y*b1y)+(c1z*c1z)),0.5);

		a2x=a[pq]-a[pq-2];
		b2y=b[pq]-b[pq-2];
		c2z=c[pq]-c[pq-2];

		if(a2x>(0.5*boxX))
			a2x=a2x-boxX;
		else
			if(a2x<-(0.5*boxX))
				a2x=boxX+a2x;

		if(b2y>(0.5*boxY))
			b2y=b2y-boxY;
		else
			if(b2y<-(0.5*boxY))
				b2y=boxY+b2y;

		if(c2z>(0.5*boxZ))
			c2z=c2z-boxZ;
		else
			if(c2z<-(0.5*boxZ))
				c2z=boxZ+c2z;
		r2=pow(((a2x*a2x)+(b2y*b2y)+(c2z*c2z)),0.5);

		a3x=a[pq]-a[pq-1];
		b3y=b[pq]-b[pq-1];
		c3z=c[pq]-c[pq-1];

		if(a3x>(0.5*boxX))
			a3x=a3x-boxX;
```

```
                        else
                              if(a3x<-(0.5*boxX))
                                    a3x=boxX+a3x;

                        if(b3y>(0.5*boxY))
                              b3y=b3y-boxY;
                        else
                              if(b3y<-(0.5*boxY))
                                    b3y=boxY+b3y;

                        if(c3z>(0.5*boxZ))
                              c3z=c3z-boxZ;
                        else
                              if(c3z<-(0.5*boxZ))
                                    c3z=boxZ+c3z;
                        r3=pow(((a3x*a3x)+(b3y*b3y)+(c3z*c3z)),0.5);

                        ex2=-2.566*(r2-1.0);

                        /*
Voh2=4.4195*pow((1-exp(ex2)),2.0);
                        Fx2=(-2.0*2.361*4.4195*(1-
exp(ex2))*(exp(ex2))*(a2x/r2));
                        Fy2=(-2.0*2.361*4.4195*(1-
exp(ex2))*(exp(ex2))*(b2y/r2));
                        Fz2=(-2.0*2.361*4.4195*(1-
exp(ex2))*(exp(ex2))*(c2z/r2));


      Vhh=0.5*aq*((r1*r1)+(r2*r2))+0.5*bq*(r3*r3)+cq*(r1+r2)*r3+dq*r1*r
2;
                        Fx3=-(a3x/r3)*(bq*r3+cq*r1+cq*r2);
                        Fy3=-(b3y/r3)*(bq*r3+cq*r1+cq*r2);
                        Fz3=-(c3z/r3)*(bq*r3+cq*r1+cq*r2);
                        */

                        Voh2=Rw*Rw*Dw*pow((r2-1.0),2.0)+0.5*bq*pow((r3-
1.6),2.0)+cq*(r1+r2-2.0)*(r3-1.6)+dq*(r1-1.0)*(r2-1.0);
                        Fx2=-(a2x/r2)*(2.0*Rw*Rw*Dw*(r2-1.0)+cq*(r3-
1.6)+dq*(r1-1.0));
                        Fy2=-(b2y/r2)*(2.0*Rw*Rw*Dw*(r2-1.0)+cq*(r3-
1.6)+dq*(r1-1.0));
                        Fz2=-(c2z/r2)*(2.0*Rw*Rw*Dw*(r2-1.0)+cq*(r3-
1.6)+dq*(r1-1.0));

                        Vhh=Rw*Rw*Dw*pow((r2-1.0),2.0)+0.5*bq*pow((r3-
1.6),2.0)+cq*(r1+r2-2.0)*(r3-1.6)+dq*(r1-1.0)*(r2-1.0);
                        Fx3=-(a3x/r3)*(bq*(r3-1.6)+cq*(r1-1.0)+cq*(r2-
1.0));
                        Fy3=-(b3y/r3)*(bq*(r3-1.6)+cq*(r1-1.0)+cq*(r2-
1.0));
                        Fz3=-(c3z/r3)*(bq*(r3-1.6)+cq*(r1-1.0)+cq*(r2-
1.0));

                        Vh=Vhh+Voh2;
                        Fxw=Fx2+Fx3;
                        Fyw=Fy2+Fy3;
```

```
                        Fzw=Fz2+Fz3;
                  }

                                                                  /*L-J
interaction within surfactant chain*/
                  for( gt=0;gt<nk;gt++)
                  {
                        if(pq==gt)
                              continue;
                        if((sele[pq]==1)||(sele[pq]==4))
                        {
                              if(sele[gt]==1||sele[gt]==4)
                              {
                                    if(lange[pq]==lange[gt])
                                    {
                                          if(gt!=(pq-1))
                                          {
                                                if(gt!=(pq-2))
                                                {
                                                      if(gt!=(pq+1))
                                                      {

      if(gt!=(pq+2))
                                                                  {

      alj=a[pq]-a[gt];

      blj=b[pq]-b[gt];

      clj=c[pq]-c[gt];


      if(alj>(0.5*boxX))

      alj=alj-boxX;
                                                                        else

      if(alj<-(0.5*boxX))

      alj=boxX+alj;


      if(blj>(0.5*boxY))

      blj=blj-boxY;
                                                                        else

      if(blj<-(0.5*boxY))

      blj=boxY+blj;


      if(clj>(0.5*boxZ))

      clj=clj-boxZ;
                                                                        else
```

```
      if(clj<-(0.5*boxZ))

      clj=boxZ+clj;


      rlj=pow(((alj*alj)+(blj*blj)+(clj*clj)),0.5);


      Vlj=2.0*0.00434*((pow((4.0/rlj),9.0))-(1.5*pow((4.0/rlj),6.0)));

      Fxlj=18.0*0.00434*alj*((pow(4.0,9.0)/pow(rlj,11.0))-
(pow(4.0,6.0)/pow(rlj,8.0)));

      Fylj=18.0*0.00434*blj*((pow(4.0,9.0)/pow(rlj,11.0))-
(pow(4.0,6.0)/pow(rlj,8.0)));

      Fzlj=18.0*0.00434*clj*((pow(4.0,9.0)/pow(rlj,11.0))-
(pow(4.0,6.0)/pow(rlj,8.0)));


      sumVlj=sumVlj+Vlj;

      sumFxlj=sumFxlj+Fxlj;

      sumFylj=sumFylj+Fylj;

      sumFzlj=sumFzlj+Fzlj;
                                                          }
                                                      }
                                                  }
                                              }
                                          }
                                      }
                                  }
                              }

      /*intermolecular interaction*/
                              for( st=0;st<neb[pq];st++)
                              {
                                    str=neighbor[pq-(myid*(nk/np))][st];
                                    ar=a[pq]-a[str];
                                    br=b[pq]-b[str];
                                    cr=c[pq]-c[str];

                                    if(ar>(0.5*boxX))
                                          ar=ar-boxX;
                                    else
                                    if(ar<-(0.5*boxX))
                                          ar=boxX+ar;
                                    if(br>(0.5*boxY))
                                          br=br-boxY;
                                    else
                                    if(br<-(0.5*boxY))
                                          br=boxY+br;
                                    if(cr>(0.5*boxZ))
                                          cr=cr-boxZ;
```

```
                        else
                        if(cr<-(0.5*boxZ))
                              cr=boxZ+cr;

                        r=pow(((ar*ar)+(br*br)+(cr*cr)),0.5);

                        if(sele[pq]==1)
      /*Head-group*/
                        {
                              qi=1.0;
                              Aii=374358.54;
                              Cii=97.07;
                        }
                        else
                        if(sele[pq]==2)
      /*Hydrogen1 of water*/
                        {
                              qi=0.41;
                              Aii=0.0;
                              Cii=0.0;
                        }
                        else
                        if(sele[pq]==3)
      /*Oxygen of water*/
                        {
                              qi=-0.82;
                              Aii=27315.25;
                              Cii=27.0;
                        }
                        else
                        if(sele[pq]==4)
      /*CH2 or CH3 of surfactant chain*/
                        {
                              qi=0.0;
                              Aii=219900.2;
                              Cii=59.69;
                        }
                        else
                        if(sele[pq]==5)
      /*Hydrogen2 of water*/
                        {
                              qi=0.41;
                              Aii=0.0;
                              Cii=0.0;
                        }
                        else
                        if(sele[pq]==7)
      /*Br-counter ion to surfactant*/
                        {
                              qi=-1.0;
                              Aii=251092.2167;
                              Cii=71.7992;
                        }
                        else
                                    if(sele[pq]==25)
/* sele[pq]==25------Si for SiO2-Warne, M. R., et al. (2000). Phys.
Chem. Chem. Phys 2: 3663-3668*/
```

```
                                {
                                        qi=0.00;
                                        Aii=0.000001;
                                        Cii=0.000001;
                                }
                                else
                                if(sele[pq]==50)
/* sele[pq]==50------O for SiO2 */
                                {
                                        qi=0.00;
                                        Aii=0.000001;
                                        Cii=0.000001;
                                }


                        if(sele[str]==1)
                        {
                              qj=1.0;
                              Ajj=374358.54;
                              Cjj=97.07;
                        }
                        else
                        if(sele[str]==2)
                        {
                              qj=0.41;
                              Ajj=0.0;
                              Cjj=0.0;
                        }
                        else
                        if(sele[str]==3)
                        {
                              qj=-0.82;
                              Ajj=27315.25;
                              Cjj=27.0;
                        }
                        else
                        if(sele[str]==4)
                        {
                              qj=0.00;
                              Ajj=219900.2;
                              Cjj=59.69;
                        }
                        else
                        if(sele[str]==5)
                        {
                              qj=0.41;
                              Ajj=0.0;
                              Cjj=0.0;
                        }
                        else
                        if(sele[str]==7)
                        {
                              qj=-1.0;
                              Ajj=251092.2167;
                              Cjj=71.7992;
                        }
                        else
```

```
                                if(sele[str]==25)
/* sele[str]==25------Si for SiO2 */
                                {
                                        qj=0.31;
                                        Ajj=197319.2193;
                                        Cjj=66.0539;
                                }
                                else
                                if(sele[str]==50)
/* sele[str]==50------O for SiO2 */
                                {
                                        qj=-0.71;
                                        Ajj=26051.5852;
                                        Cjj=26.4645;
                                }



                        Aij=pow((Aii*Ajj),0.5);
                        Cij=pow((Cii*Cjj),0.5);


                        if(sele[pq]==4||sele[pq]==1)
                        {
                              if(sele[str]==4||sele[str]==1)
                              {
                                    if(lange[pq]==lange[str])
                                    {
                                          continue;

                                    }
                                    else
      /*Chain-Chain interaction of two different surfactants*/
                                    {

      V12=2.0*0.00434*((pow((4.0/r),9.0))-(1.5*pow((4.0/r),6.0)));

      Fx12=18.0*0.00434*ar*((pow(4.0,9.0)/pow(r,11.0))-
(pow(4.0,6.0)/pow(r,8.0)));

      Fy12=18.0*0.00434*br*((pow(4.0,9.0)/pow(r,11.0))-
(pow(4.0,6.0)/pow(r,8.0)));

      Fz12=18.0*0.00434*cr*((pow(4.0,9.0)/pow(r,11.0))-
(pow(4.0,6.0)/pow(r,8.0)));

                                          sumV12=sumV12+V12;
                                          sumFx12=sumFx12+Fx12;
                                          sumFy12=sumFy12+Fy12;
                                          sumFz12=sumFz12+Fz12;


                                          Aij=0.0;
                                          Cij=0.0;
                                    }
                              }
                        }
```

```
                        if(sele[pq]==2||sele[pq]==3||sele[pq]==5)

                        {

      if(sele[str]==2||sele[str]==3||sele[str]==5)
                              {
                                    if(lange[pq]==lange[str])
                                          continue;
                              }
                        }


                        KRIJ=KAPPA*r;

                            TE=1.0/(1.0+P*KRIJ);
                                KRIJSQ=KRIJ*KRIJ;


TEP=TE*(A1+TE*(A2+TE*(A3+TE*(A4+TE*A5))));

                                    ERFC=TEP*exp(-KRIJSQ);


      Vr=(0.5*14.3979084*qi*qj*ERFC/r)+(Aij/pow(r,12.0))-
(Cij/pow(r,6.0));             /*Coulombic potential + L-J potential*/

                        if(qi==0||qj==0)
                        {
                              FRX=0.0;
                                        FRY=0.0;
                                        FRZ=0.0;
                        }
                        else
      /*Force calculations of Coulombic potential with Ewald
summation*/
                        {

      FRX=0.5*14.3979084*qi*qj*(ERFC/r+((2.0*KAPPA/1.77)*(exp(-
(KAPPA*KAPPA*r*r)))))*(ar/pow(r,2.0));

FRY=0.5*14.3979084*qi*qj*(ERFC/r+((2.0*KAPPA/1.77)*(exp(-
(KAPPA*KAPPA*r*r)))))*(br/pow(r,2.0));

FRZ=0.5*14.3979084*qi*qj*(ERFC/r+((2.0*KAPPA/1.77)*(exp(-
(KAPPA*KAPPA*r*r)))))*(cr/pow(r,2.0));

                        }
                        Fx=FRX+(12.0*Aij*ar/pow(r,14.0))-
(6.0*Cij*ar/pow(r,8.0));            /*Force calculation of L-J
potential*/
                        Fy=FRY+(12.0*Aij*br/pow(r,14.0))-
(6.0*Cij*br/pow(r,8.0));
                        Fz=FRZ+(12.0*Aij*cr/pow(r,14.0))-
(6.0*Cij*cr/pow(r,8.0));


                        sumVr=sumVr+Vr;
```

```
                        sumforcex=Fx+sumforcex;
                        sumforcey=Fy+sumforcey;
                        sumforcez=Fz+sumforcez;

                  }

                  /*Coulubic term in K-space*/
                  TOTK=0;
      /*
                  if(qi!=0)
                  {
                        for(KX=-KMAX;KX<=KMAX;KX++)
                        {
                              for(KY=-KMAX;KY<=KMAX;KY++)
                              {
                                    for(KZ=-KMAX;KZ<=KMAX;KZ++)
                                    {

                                    KSQ=KX*KX+KY*KY+KZ*KZ;

      if((KSQ<KSQMAX)&&(KSQ!=0))
                                                {
                                                      TOTK=TOTK+1;
                                          cosx=sinx=0.0;
                                          for (t=0;t<neb[pq];t++)
                                          {
                                                str=neighbor[pq-
(myid*(nk/np))][t];

      if(lange[pq]==lange[str])
                                                      continue;

                                          if (sele[str]==1)
                                                qj=1.0;
                                          if
(sele[str]==2||sele[str]==5)
                                                qj=0.41;
                                          if (sele[str]==4)
                                          {
                                                qj=0.0;
                                                continue;
                                          }
                                          if (sele[str]==3)
                                                qj=-0.82;
                                          if (sele[str]==7)
                                                qj=-1.0;
                                          if(sele[str]==6)
                                          {
                                                qj=0.0;
                                                continue;
                                          }


      cosx=cosx+qj*cos(TWOPI/boxX*KX*a[str]+TWOPI/boxY*KY*b[str]+TWOPI/
boxZ*KZ*c[str]);
```

```
      sinx=sinx+qj*sin(TWOPI/boxX*KX*a[str]+TWOPI/boxY*KY*b[str]+TWOPI/
boxZ*KZ*c[str]);
                                          }


      VK=(1/(2*e0*Vol))*KVEC[TOTK]*(pow(cosx,2.0)+pow(sinx,2.0));

      temp1=sin(TWOPI/boxX*KX*a[pq]+TWOPI/boxY*KY*b[pq]+TWOPI/boxZ*KZ*c
[pq]);

      temp2=cos(TWOPI/boxX*KX*a[pq]+TWOPI/boxY*KY*b[pq]+TWOPI/boxZ*KZ*c
[pq]);


      FKX=1/(e0*Vol)*qi*KVEC[TOTK]*(TWOPI*KX/boxX)*(temp1*cosx-
temp2*sinx);

      FKY=1/(e0*Vol)*qi*KVEC[TOTK]*(TWOPI*KY/boxY)*(temp1*cosx-
temp2*sinx);

      FKZ=1/(e0*Vol)*qi*KVEC[TOTK]*(TWOPI*KZ/boxZ)*(temp1*cosx-
temp2*sinx);

                                          sumVK=VK+sumVK;

                                          sumFKX=sumFKX+FKX;
                                          sumFKY=sumFKY+FKY;
                                          sumFKZ=sumFKZ+FKZ;
                                    }
                              }
                        }
                  }
                  }
      */
                  if(sele[pq]==1)
      /*Total potential energy and acceleration calculation for each
particle*/
                  {
                        V=Vs+Vb+Vt+sumVlj+sumV12+sumVr+sumVK;

      accx=9648.85*(Fxs+sumFxlj+sumFx12+sumforcex+sumFxsur+sumFxsili+Fx
b+Fxt+sumFKX)/mass;

      accy=9648.85*(Fys+sumFylj+sumFy12+sumforcey+sumFysur+sumFysili+Fy
b+Fyt+sumFKY)/mass;

      accz=9648.85*(Fzs+sumFzlj+sumFz12+sumforcez+sumFzsur+sumFzsili+Fz
b+Fzt+sumFKZ)/mass;

                  }
                  if(sele[pq]==2||sele[pq]==5)
                  {
                        V=Vh+sumVr+sumVK;

      accx=9648.85*(Fxw+sumforcex+sumFxsur+sumFxsili+sumFKX);
```

```
      accy=9648.85*(Fyw+sumforcey+sumFysur+sumFysili+sumFKY);

      accz=9648.85*(Fzw+sumforcez+sumFzsur+sumFzsili+sumFKZ);
                  }
                  if(sele[pq]==3)
                  {
                        V=Vo+sumVr+sumVK;

      accx=9648.85*(Fxw+sumforcex+sumFxsur+sumFxsili+sumFKX)/mass;

      accy=9648.85*(Fyw+sumforcey+sumFysur+sumFysili+sumFKY)/mass;

      accz=9648.85*(Fzw+sumforcez+sumFzsur+sumFzsili+sumFKZ)/mass;

                  }
                  if(sele[pq]==4)
                  {
                        V=Vs+Vb+Vt+sumVlj+sumV12+sumVr+sumVK;

      accx=9648.85*(Fxs+sumFxlj+sumFx12+sumforcex+sumFxsur+sumFxsili+Fx
b+Fxt+sumFKX)/mass;

      accy=9648.85*(Fys+sumFylj+sumFy12+sumforcey+sumFysur+sumFysili+Fy
b+Fyt+sumFKY)/mass;

      accz=9648.85*(Fzs+sumFzlj+sumFz12+sumforcez+sumFzsur+sumFzsili+Fz
b+Fzt+sumFKZ)/mass;
                  }
                  if(sele[pq]==7)
                  {
                        V=sumVr+sumVK;
                        accx=9648.85*(sumforcex+sumFKX)/mass;
                        accy=9648.85*(sumforcey+sumFKY)/mass;
                        accz=9648.85*(sumforcez+sumFKZ)/mass;
                  }
                  if(sele[pq]==25)
                        {
                        V=0.000001;
                                accx=0.00001;
                                accy=0.00001;
                                accz=0.00001;
                        }
                        if(sele[pq]==50)
                        {
                        V=0.000001;
                                accx=0.00001;
                                accy=0.00001;
                                accz=0.00001;
                        }

                  acclx[ps][pq]=accx;
                  accly[ps][pq]=accy;
                  acclz[ps][pq]=accz;

                  if(ps==1)
                  {
```

```
                        ts=ps-1;
                }
                else
                {
                        ts=ps+1;
                }

                if((tstep==0))
      /*Inital assignment of velocities to each particle*/
                {
      /*                if(sele[pq]==1||sele[pq]==7)
                        {
                              vx[ps][pq]=0.0001;
                              vy[ps][pq]=0.0001;
                              vz[ps][pq]=0.0001;
                        }
                        else
                                if(sele[pq]==25||sele[pq]==50)
                                {

                                      vx[ps][pq]=0.00001;
                                        vy[ps][pq]=0.00001;
                                        vz[ps][pq]=0.00001;

                                }
                        else
                        {
                              vx[ps][pq]=0.0001;
                              vy[ps][pq]=0.0001;
                              vz[ps][pq]=0.0001;
                        }
      */
                }
                else
      /*Velocity calculation*/
                {

      vx[ps][pq]=(vx[ts][pq]+(double)((0.5)*delta*(acclx[ts][pq]+acclx[
ps][pq])))*pow(1.0+(0.1*(((nk*temp)/TKE)-1.0)),0.5);

      vy[ps][pq]=(vy[ts][pq]+(double)((0.5)*delta*(accly[ts][pq]+accly[
ps][pq])))*pow(1.0+(0.1*(((nk*temp)/TKE)-1.0)),0.5);

      vz[ps][pq]=(vz[ts][pq]+(double)((0.5)*delta*(acclz[ts][pq]+acclz[
ps][pq])))*pow(1.0+(0.1*(((nk*temp)/TKE)-1.0)),0.5);
                }

                if(tstep==(time-1))
                {
                        Velox[pq]=vx[ps][pq];
                        Veloy[pq]=vy[ps][pq];
                        Veloz[pq]=vz[ps][pq];
                        Accox[pq]=acclx[ps][pq];
                        Accoy[pq]=accly[ps][pq];
                        Accoz[pq]=acclz[ps][pq];
                }
```

```
      Vel[ps][pq]=pow((pow(vx[ps][pq],2.0)+pow(vy[ps][pq],2.0)+pow(vz[p
s][pq],2.0)),0.5);

                  T[ps][pq]=(11606.3*mass*pow(Vel[ps][pq],2.0));


                  PE=PE+V;                              /*Potential
energy calculation*/
                  KE=KE+T[ps][pq];                      /*Kinetic energy
calculation*/


      x1new=a[pq]+(delta*vx[ps][pq])+((0.5)*delta*delta*acclx[ps][pq]);
      /*New postion-velocity Verlet method*/

      y1new=b[pq]+(delta*vy[ps][pq])+((0.5)*delta*delta*accly[ps][pq]);

      z1new=c[pq]+(delta*vz[ps][pq])+((0.5)*delta*delta*acclz[ps][pq]);

                  anew[pq]=x1new;
                  bnew[pq]=y1new;
                  cnew[pq]=z1new;

                  if(anew[pq]<-(boxX/2.0))                  /*Periodic
Boundary Condition calculations*/
                        anew[pq]=anew[pq]+boxX;
                  else
                        if(anew[pq]>(boxX/2.0))
                              anew[pq]=anew[pq]-boxX;
                  if(bnew[pq]<-(boxY/2.0))
                        bnew[pq]=bnew[pq]+boxY;
                  else
                        if(bnew[pq]>(boxY/2.0))
                              bnew[pq]=bnew[pq]-boxY;
                  if(cnew[pq]<-(boxZ/2.0))
                        cnew[pq]=cnew[pq]+boxZ;
                  else
                        if(cnew[pq]>(boxZ/2.0))
                              cnew[pq]=cnew[pq]-boxZ;

            }

            ps++;

            st2 = MPI_Wtime(); /* JAIN - Start time for barrier
synchronization  */

            MPI_Barrier(MPI_COMM_WORLD); /*     JAIN -This is a
      barrier     synchronization   call. All 'np' processors compute
the above   'for loop' and wait     here till all of them have finished
      */

            et2 = MPI_Wtime(); /* JAIN - End time for Barrier
synchronization  */
```

```
            tt2 = tt2 + (et2-st2); /* JAIN - Total time for barrier
synchronization  */


            /* JAIN     -The first processor acts as the master   node
and the others    are   slave nodes */

            st3 = MPI_Wtime(); /* JAIN - Start time for total
communication  */
            st4 = MPI_Wtime(); /* JAIN - Start time for slave nodes
sending the x,y,z coordinates  */


            if(myid!=0) /* JAIN     - If not the master     node then
enter this 'if loop' */
            {
                  /* JAIN     - Send the x-coordinates of   all   your
particles to the master node */
                  MPI_Send(anew,nk,MPI_DOUBLE,0,0,MPI_COMM_WORLD);
                  /* JAIN     - Send the y-coordinates of   all   your
particles to the master node */
                  MPI_Send(bnew,nk,MPI_DOUBLE,0,1,MPI_COMM_WORLD);
                  /* JAIN     - Send the z-coordinates of   all   your
particles to the master node */
                  MPI_Send(cnew,nk,MPI_DOUBLE,0,2,MPI_COMM_WORLD);

                  /* JAIN     - Send the sum of potential energies for
all your nk/np paticles to the master node */
                  MPI_Send(&PE,1,MPI_DOUBLE,0,6,MPI_COMM_WORLD);
                  /* JAIN     - Send the sum of temperatures of all
your nk/np paticles to the master node */
                  MPI_Send(&KE,1,MPI_DOUBLE,0,7,MPI_COMM_WORLD);

                  if(tstep==(time-1))
                    {
                      MPI_Send(Velox,nk,MPI_DOUBLE,0,9,MPI_COMM_WORLD);

MPI_Send(Veloy,nk,MPI_DOUBLE,0,10,MPI_COMM_WORLD);

MPI_Send(Veloz,nk,MPI_DOUBLE,0,11,MPI_COMM_WORLD);


MPI_Send(Accox,nk,MPI_DOUBLE,0,12,MPI_COMM_WORLD);

MPI_Send(Accoy,nk,MPI_DOUBLE,0,13,MPI_COMM_WORLD);

MPI_Send(Accoz,nk,MPI_DOUBLE,0,14,MPI_COMM_WORLD);
                    }
            }

            et4 = MPI_Wtime(); /* JAIN - End time for slave nodes
sending the x,y,z coordinates  */
            tt4 = tt4 + (et4-st4); /* JAIN - Total time for slave nodes
sending the x,y,z coordinates  */
            st5 = MPI_Wtime(); /* JAIN - Start time for the master node
receiving the x,y,z coordinates  */
```

```
            if(myid==0)  /*   JAIN - If master node then enter this
'else loop' */
            {
              TPE = PE;
              TKE = KE;

                  for(i=1;i<np;i++) /* JAIN -   For   all   1 to np-1
slave nodes */
                  {
                        /* JAIN     - Receive the new x-coordinates
      of all particles from slave   node i */

      MPI_Recv(a1new,nk,MPI_DOUBLE,i,0,MPI_COMM_WORLD,&status);
                        /* JAIN     - Receive the new y-coordinates
      of all particles from slave   node i */

      MPI_Recv(b1new,nk,MPI_DOUBLE,i,1,MPI_COMM_WORLD,&status);
                        /* JAIN     - Receive the new z-coordinates
      of all particles from slave   node i */

      MPI_Recv(c1new,nk,MPI_DOUBLE,i,2,MPI_COMM_WORLD,&status);

                        /* JAIN     - Receive the sum of potential
energies of all the particles of slave i */

      MPI_Recv(&PEnew,1,MPI_DOUBLE,i,6,MPI_COMM_WORLD,&status);
                        /* JAIN     - Receive the sum of temperatures
for all the particles of slave i */

      MPI_Recv(&KEnew,1,MPI_DOUBLE,i,7,MPI_COMM_WORLD,&status);

                       if(tstep==(time-1))
                         {

MPI_Recv(Veloxnew,nk,MPI_DOUBLE,i,9,MPI_COMM_WORLD,&status);

MPI_Recv(Veloynew,nk,MPI_DOUBLE,i,10,MPI_COMM_WORLD,&status);

MPI_Recv(Veloznew,nk,MPI_DOUBLE,i,11,MPI_COMM_WORLD,&status);


MPI_Recv(Accoxnew,nk,MPI_DOUBLE,i,12,MPI_COMM_WORLD,&status);

MPI_Recv(Accoynew,nk,MPI_DOUBLE,i,13,MPI_COMM_WORLD,&status);

MPI_Recv(Accoznew,nk,MPI_DOUBLE,i,14,MPI_COMM_WORLD,&status);
                          }

                       TPE = TPE + PEnew;
                       TKE = TKE + KEnew;

                       /* JAIN     - Update the particle coordinates
for the nk/np     particles sent by slave node i */

      for(j=i*(nk/np);j<(i*(nk/np)+(nk/np)+mod[i]);j++)
                       {
```

```
                              anew[j]=a1new[j]; /* JAIN -   'x'
      coordinate */
                              bnew[j]=b1new[j]; /* JAIN -   'y'
      coordinate */
                              cnew[j]=c1new[j]; /* JAIN -   'z'
      coordinate */
                        }


                        et5 = MPI_Wtime();  /* JAIN - End time for the
master node receiving the x,y,z coordinates  */
                        tt5 = tt5 + (et5-st5);  /* JAIN - Total time
for the master node receiving the x,y,z coordinates  */



                        if(tstep==(time-1))
                          {

for(j=i*(nk/np);j<(i*(nk/np)+(nk/np)+mod[i]);j++)
                              {
                              Velox[j]=Veloxnew[j]; /* JAIN -     'x'
      coordinate */
                              Veloy[j]=Veloynew[j]; /* JAIN -     'y'
      coordinate */
                              Veloz[j]=Veloznew[j]; /* JAIN -     'z'
      coordinate */

                              Accox[j]=Accoxnew[j]; /* JAIN -     'x'
      coordinate */
                              Accoy[j]=Accoynew[j]; /* JAIN -     'y'
      coordinate */
                              Accoz[j]=Accoznew[j]; /* JAIN -     'z'
      coordinate */

                              }
                          }
                  }

                  KE=(0.5*TKE)/(1.5*9648.85);
                  TKE = KE;

                  if(tstep==1000*count)                    /*Print
statement of timestep, Potential energy and Temperature of the system-
'e'nergy file*/
                  {
                    count++;
                    fprintf(fq,"%d  %30.12lf
%30.12lf\n",tstep,(TPE/nk),(TKE/nk));
                  }

                  /*EK = TEK;*/

                  for( it=0;it<nk;it++)
                  {
                        a[it]=anew[it];
                        b[it]=bnew[it];
```

```
                        c[it]=cnew[it];

                }
                if(tstep==(time-1))
                {
                        fprintf(fr,"%d  1   300   0.001\n",nk);
/*Configuration output*/

                        plus=0;
                        for(it=0;it<nk;it++)
                        {
                                fprintf(fr,"%d   %d   %lf %lf %lf
%d\n",++plus,sele[it],a[it],b[it],c[it],lange[it]);
                        }
                        plus=0;
                        for(it=0;it<nk;it++)
                        {
                                fprintf(fr,"%d   %d   %lf %lf %lf
%d\n",++plus,sele[it],Velox[it],Veloy[it],Veloz[it],lange[it]);
                        }
                        plus=0;
                        for(it=0;it<nk;it++)
                        {
                                fprintf(fr,"%d   %d   %lf %lf %lf
%d\n",++plus,sele[it],Accox[it],Accoy[it],Accoz[it],lange[it]);
                        }
                }

                /* JAIN     - The master node broadcasts the new
x,y,z coordinates of all 'nk' particles to      all   'np' slave nodes
(processors) */

      /*          MPI_Bcast(a,nk,MPI_DOUBLE,0,MPI_COMM_WORLD);
                  MPI_Bcast(b,nk,MPI_DOUBLE,0,MPI_COMM_WORLD);
                  MPI_Bcast(c,nk,MPI_DOUBLE,0,MPI_COMM_WORLD);*/
            }

            if(myid==0) /* JAIN - If master node then enter this 'if
loop' */
                  {

                st6 = MPI_Wtime();  /* JAIN - Start time for the master
node to send broadcast  */

                for(i=1;i<np;i++) /* JAIN - For all 1 to np-1 slave
nodes */
                   {
                    /* JAIN - Send the x-coordinates of all your
particles to the master node */
                    MPI_Send(a,nk,MPI_DOUBLE,i,3,MPI_COMM_WORLD);
                    /* JAIN - Send the y-coordinates of all your
particles to the master node */
                    MPI_Send(b,nk,MPI_DOUBLE,i,4,MPI_COMM_WORLD);
                    /* JAIN - Send the z-coordinates of all your
particles to the master node */
                    MPI_Send(c,nk,MPI_DOUBLE,i,5,MPI_COMM_WORLD);
```

```
                MPI_Send(&TKE,1,MPI_DOUBLE,i,8,MPI_COMM_WORLD);

                   }
                et6 = MPI_Wtime();  /* JAIN - End time for the master
node to send broadcast  */
                tt6 = tt6 + (et6-st6);  /* JAIN - Total time for the
slave nodes to receive broadcast  */

                plus=0;
                 }

             else /* JAIN - If NOT master node then enter this 'else
loop' */
                  {
                st7 = MPI_Wtime();  /* JAIN - Start time for the slave
nodes to receive broadcast  */

                        /* JAIN - Receive the new x-coordinates of all
particles from slave node i*/

MPI_Recv(a,nk,MPI_DOUBLE,0,3,MPI_COMM_WORLD,&status);
                           /* JAIN - Receive the new y-coordinates of
all particles from slave node i*/

MPI_Recv(b,nk,MPI_DOUBLE,0,4,MPI_COMM_WORLD,&status);
                           /* JAIN - Receive the new z-coordinates of
all particles from slave node i*/

MPI_Recv(c,nk,MPI_DOUBLE,0,5,MPI_COMM_WORLD,&status);


MPI_Recv(&TKE,1,MPI_DOUBLE,0,8,MPI_COMM_WORLD,&status);
                      et7 = MPI_Wtime();  /* JAIN - End time for the
slave nodes to receive broadcast  */
                      tt7 = tt7 + (et7-st7);  /* JAIN - Total time for
the slave nodes to receive broadcast  */

             }

            et3 = MPI_Wtime();  /* JAIN - End time for total
communication  */
            tt3 = tt3 + (et3-st3);  /* JAIN - Total time for total
communication  */


            /* JAIN     - End of modifications */

      }


      /* JAIN - Begin modifications */

      if (myid <=np )
        {
          printf("Time to barrier synch for processor %d is : %30.15lf
\n",myid, tt2); /* Master node prints barrier synch time */
        }
```

```
      if (myid == 0)
        {
                printf("Total communication time: %30.15lf \n", tt3);
/* Master node prints total communication time */
        }

      if ((np > 1) && (myid == 1))
        {
          printf("Slave time to send x,y,z: %30.15lf \n", tt4); /*
Slave node 1 prints time to send x,y,z coordinates to master node */
        }

      if ((np > 1) && (myid == 0))
        {
          printf("Master time to receive x,y,z: %30.15lf \n", tt5); /*
Master node prints time to recieve x,y,z coordinates from slaves */
        }

      if ((np > 1) && (myid == 0))
        {
                printf("Master time to send broadcast: %30.15lf \n",
tt6); /* Master node prints broadcast send time */
        }

      if ((np > 1) && (myid == 1))
        {
          printf("Slave time to receive broadcast: %30.15lf \n", tt7);
/* Slave node 1 prints broadcast recieve time */
        }

      et8 = MPI_Wtime();
      tt8 = et8 - st8;

      if (myid<np)
        {
          printf ("Total execution time for processor %d is: %30.15lf
\n", myid, tt8);
        }

      MPI_Finalize();   /* JAIN     - End of MPI system     */
}

/* JAIN     - Begin     Modifications */

double      cosd(double d)
{
      double r=0.0;
      r=d*(22.0/7.0)/180.0;
      return(cos(r));
}


/* JAIN     - End of modifications */
```

LIST OF REFERENCES

1. S. Manne, H. E. Gaub *Science* **1995**, *270*, 1480.

2. P. K. Singh, J. J. Adler., Y. I. Rabinovich, B. M. Moudgil *Langmuir* **2001**, *17*, 468-473.

3. Y. I. Rabinovich, I. U. Vakarelski, Scott C. Brown, P. K. Singh, B. M. Moudgil *J. Colloid Interf.Sci.* **2004**, *270*, 29-36.

4. J. J. Adler, P. K. Singh., A. Patist, Y. Rabinovich, D. O. Shah, B. M. Moudgil *Langmuir* **2000**, *16*, 7255-7262.

5. J. N. Israelachvili *Intermolecular and Surface Forces* London: Academic Press, 1992.

6. B. M. Moudgil, H. Soto, P. Somasundaran In *Reagents in Mineral Technology*; P. Somasundaran, B. M. Moudgil., Ed., p 79 NY: Marcel Dekker, 1988.

7. *Surfactant Solutions: New Methods of Investigations*; Marcel Dekker: New York, 1987.

8. P. Lianos, R. Zana. *J. Coll. Interface Sci.* **1981**, *84*, 100.

9. G. Biresaw, D. McKenzie, C. Bunton, and D. Nicoli *J. Phys. Chem.* **1985**, *89*, 5144.

10. D. McKenzie, C. Bunton, D. Nicoli, and G. Savelli *J. Phys. Chem.* **1987**, *91*, 5709.

11. R. Dorshow, C. Bunton, and A. Nicoli *J. Phys. Chem.* **1983**, *87*, 1409.

12. A. Malliaris, J. Le Moigne, J. Sturn, and R. Zana *J. Phys. Chem.* **1985**, *89*, 2709.

13. E. Roelants, F. C. De Schryver *Langmuir* **1987**, *3*, 209.

14. E. Roelants, E. Gelade, J. Smid, and F. C. De Schryver *J. Coll. Interface Sci.* **1985**, *107*, 337.

15. H. Rupprecht *J. Pharm. Sci.* **1972**, *61*, 700.

16. H. Walderhaug, O. Soderman, and P. Stilbs *J. Phys. Chem.* **1984**, *88*, 1755.

17. J. Tabony *Mol. Phys.* **1984**, *51*, 974.

18. F. Reiss-Husson, V. Luzzatti *J. Phys. Chem.* **1964**, *68*, 3504.

19. P. V. Coveney, J. A. D. Wattis *Proc. R. Soc. London, Ser. A* **1996**, *452*, 2079.

20. J. B. Maillet, V. Lachet, Peter V. Coveney *Phys.Chem.Chem.Phys.* **1999**, *1*, 5277-5290.

21. A. Fan, P. Somasundaran, N.J. Turro *Langmuir* **1997**, *13*, 506.

22. T. Gu, Z. Huang *Colloids Surf.* **1989**, *40*, 71.

23. J. H. Harwell, J. C. Hoskins, R. S. Schechter, W. H. Wade *Langmuir* **1985**, *1*, 251.

24. Y. Gao, J. Du, T. Gu *J. Chem. Soc. Faraday Trans.* **1987**, *1*, 2671.

25. H. Rupprecht, E. Ullmann, K. Thoma *Fortsch. Kolloid Polym.* **1971**, *55*, 45.

26. H. Rupprecht, T. Gu *Colloid Polym. Sci.* **1991**, *269*, 506.

27. B. H. Bijsterbosch *J. Colloid Interf.Sci.* **1974**, *47*, 186.

28. P. Somasundaran, D. W. Fuerstenau *J. Phys. Chem.* **1966**, *70*, 90.

29. P. Chandar, P. Somasundaran, N.J. Turro *J. Colloid Interf.Sci.* **1987**, *117*, 148.

30. P.Somasundaran, J. T. Kunjappu, C.V.Kumar, N.J. Turro, J.K. Barton *Langmuir* **1989**, *5*, 215.

31. R. Atkin, V. S. J. Craig, E.J. Wainless, S. Biggs *Advances in Colloid and Interface Science* **2003**, *103*, 219-304.

32. H. N. Patrick, G. G. Warr, S. Manne, I. A. Aksay *Langmuir* **1997**, *13*, 4349.

33. S. Manne, J. P. Cleveland, H. E. Gaub, G. D. Stucky, P. K. Hansma *Langmuir* **1994**, *10*, 4409.

34. A. M. Koganovkii *Colloid J. USSR* **1962**, *24*, 702.

35. S. Manne, T. E. Schaffer, Q. Huo, P. K. Hansma, D. E. Morse, G. D. Stucky, I. A. Aksay *Langmuir* **1997**, *13*, 6382.

36. E. J. Wanless, W. A. Ducker *J. Phys. Chem. B* **1996**, *100*, 3207.

37. E. J. Wanless, W. A. Ducker *Langmuir* **1997**, *13*, 1463.

38. L. M. Grant, F. Tiberg, W. A. Ducker *J. Phys. Chem. B* **1998**, *102*, 4288.

39. N. B. Holland, M. Ruegsegger, R. E. Marchant *Langmuir* **1998**, *14*, 2790.

40. W. A. Ducker, L. M. Grant *J. Phys. Chem. B* **1996**, *100*, 11507.

41. W. A. Ducker, E. G. Wanless *Langmuir* **1996**, *12*, 5915.

42. W. A. Ducker, E. G. Wanless *Langmuir* **1999**, *15*, 160.

43. S.B. Velegol, B. D. Fleming, S. Biggs, E.J. Wanless, R.D. Tilton *Langmuir* **2000**, *16*, 2548.

44. J. -F. Liu, G. Min, W. A. Ducker *Langmuir* **2001**, *17*, 4895.

45. V. Subramaniam, W. A. Ducker *Langmuir* **2000**, *16*, 4447.

46. P. Levitz, H. V. Damme, D. Keravis *J. Phys. Chem.* **1984**, *88*, 2228.

47. P. Levitz, H. V. Damme *J.Phys.chem.* **1986**, *90*, 1302.

48. C. Strom, P. Hansson, B. Jonsson, O. Soderman *Langmuir* **2000**, *16*, 2469.

49. A.R. Rennie, E. M. Lee, E.A. Simister, R.K. Thomas *Langmuir* **1990**, *6*, 1031.

50. G. Fragneto, R. K. Thomas, A.R. Rennie, J. Penfold *Langmuir* **1996**, *12*, 6036.

51. D.C. McDermott, J. R. Lu, E.M. Lee, R.K. Thomas, A.R. Rennie *Langmuir* **1992**, *8*, 1204.

52. D. C. McDermott, J. McCarney, R. K. Thomas, A. R. Rennie *J. Coll. Interface Sci.* **1994**, *162*, 304.

53. J. C. Schulz, G. G. Warr, W. A. Hamilton, P. D. Butler *J. Phys. Chem. B* **1999**, *103*, 11057.

54. J. C. Schulz, G. G. Warr, P. D. Butler, W. A. Hamilton *Phys. Rev. E* **2001**, *63*, 604.

55. R. M. A. Azzam, N. M. Bashara *Ellipsometry and Polarized light* Amsterdam: North-Holland Personal Library, 1989.

56. J. C. Dijt, M. A. Cohen Stuart, G. J. Fleer *Adv. Colloid Interf. Sci.* **1994**, *50*, 79.

57. K. Eskilsson, V. V. Yaminsky *Langmuir* **1998**, *14*, 2444.

58. E. S. Pagac, D. C. Prieve, R. D. Tilton *Langmuir* **1998**, *14*, 2444.

59. R. Atkin, V. S. J. Craig, S. Biggs *Langmuir* **2000**, *16*, 9374.

60. P. Somasundaran, J. T. Kunjappu *Colloids Surf.* **1989**, *37*, 245.

61. P. Chandar, P. Somasundaran, N. J. Turro *J. Colloid Interf.Sci.* **1987**, *117*, 31.

62. G. Fragneto, R. K. Thomas, A. R. Rennie, J. Penfold *Langmuir* **1996**, *12*, 6036-6043.

63. G.J. Martina, M. E. Tuckerman, D.J. Tobias, M.L. Klein *Mol. Phys.* **1996**, *87*, 1117.

64. M. Tarek, S. Bandyopadhyay, M.L. Klein *J. Mol. Liq.* **In press**.

65. S. Bandyopadhyay, J. C. Shelly, M. Tarek, P.B. Moore, M. L. Klein *J.Phys.Chem.B.* **1998**, *102*, 6318-6322.

66. B. J. Alder, T. E. Wainwright *Journal of Chemical Physics* **1957**, *27*, 1208-1209.

67. B. J. Alder, T. E. Wainwright *Journal of Chemical Physics* **1959**, *31*, 459-466.

68. M. P. Allen, D. J. Tildesley *Computer Simulation of Liquids* Oxford: Clarendon, 1987.

69. R. A. Alberty *Physical Chemistry* 7 ed. Singapore: John Willey & Sons, 1987.

70. S.Nose *Molecular Physics* **1984**, *52*, 255-268.

71. W. G. Hoover *Physical Review A* **1985**, *31*, 1695-1697.

72. H.J.C. Berendsen, J. P. M. Postma, W.F. Vangunsteren, A. Dinola, J.R. Haak *J. Chem. Phys.* **1984**, *81*, 3684-3690.

73. H. C. Anderson *J. Chem. Phys.* **1980**, *72*, 2384-2393.

74. D. Frenkel, B. Smit *Understanding Molecular Simulation* San Diego: Academic Press, 1996.

75. W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson *Journal of Chemical Physics* **1982**, *76*, 637-649.

76. P. Ewald *Ann. Phys.* **1921**, *64*, 253-287.

77. E. Madelung *Phys. Z.* **1918**, *19*, 524-532.

78. T. A. Weber *J. Chem. Phys* **1978**, *69*, 2347-2354.

79. K. Toukan, A. Rahman *Phys. Rev. B.* **1985**, *94*, 2643-2648.

80. M. R. Warne, N. L. Allan, T. Cosgrove *Phys. Chem. Chem. Phys* **2000**, *2*, 3663-3668.

81. S. L. Lin, J. Mellor-Crummey., B. M. Pettitt, Jr. G. N. Phillips *Journal of Computational Chemistry* **1992**, *13*, 1022-1035.

82. T. Imae, S. Ikeda *J. Chem. Phys* **1986**, *90*, 5216.

83. J. Jacobsen, B. H. Cooper, J.P. Sethna *Phys. Rev. B* **1998**, *58*, 15847-15865.

84. M.R. Sorensen, and A. F. Voter *J. Chem. Phys.* **2000**, *112*, 9599-9606.

BIOGRAPHICAL SKETCH

Kunal Shah was born on October 22nd, 1978, in Ahmedabad, Gujarat, India. His father is a stock investor and mother is a homemaker. He finished his high school education from Sharda Mandir Vinay Mandir in 1996. He enrolled in L.D. College of Engineering, Ahmedabad, for his Bachelor of Engineering degree with specialization in plastics technology. He graduated with his bachelor's degree in June 2000. In August 2000, he came to the United States in pursuit of graduate education and was admitted to the Department of Materials Science and Engineering at the University of Florida. He joined Dr. Sinnott's group for his Ph.D. research work. He expects to obtain his Doctor of Philosophy degree in May 2005. He plans to join Intel at Portland, Oregon.