4-19-2013

THE ROLE OF MOLECULAR DYNAMICS SIMULATIONS IN INVESTIGATING DIFFUSION ON MESOSCOPIC SCALES

Santona Tuli
Trinity University, stuli@trinity.edu

THE ROLE OF MOLECULAR DYNAMICS SIMULATIONS IN INVESTIGATING DIFFUSION ON MESOSCOPIC SCALES

SANTONA TULI

A DEPARTMENT HONORS THESIS SUBMITTED TO THE
DEPARTMENT OF PHYSICS AND ASTRONOMY AT TRINITY UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR GRADUATION WITH
DEPARTMENTAL HONORS

DATE 19th APRIL 2013

_____________________________ _________________________________
THESIS ADVISOR DEPARTMENT CHAIR

____
ASSOCIATE VICE PRESIDENT FOR ACADEMIC AFFAIRS,
CURRICULUM AND STUDENT ISSUES

TABLE OF CONTENTS

Abstract 3

Acknowledgements 4

I. Introduction 5

II. Classical Transport Theory: *Brownian Motion for the Macroscopic Scale* 9

III. The Mesoscopic Scale 13

a. Velocity Autocorrelation 14

b. The Nature of Mesoscopic Diffusion 18

IV. Simulations and their Results 21

a. Discussion of GROMACS Simulation 21

b. Discussion of MATLAB Simulation 24

V. Conclusion 33

References 35

Appendix A 37

Appendix B 41

Santona Tuli. Honors Thesis, Spring 2013.
Trinity University Department of Physics and Astronomy.

THE ROLE OF MOLECULAR DYNAMICS SIMULATIONS IN INVESTIGATING DIFFUSION ON MESOSCOPIC SCALES

Abstract: *In order to investigate transport properties of molecular solutions on mesoscopic scales, we use the fluctuation-dissipation theorem and velocity and noise autocorrelation to determine the diffusion constant of two simulated solutions of particles interacting through Lennard-Jones potentials. This thesis describes classical transport theories which are valid for macroscopic diffusion, and includes a discussion of the nature of the force on solute particles which are comparable in size to solvent particles (we call diffusion in this limit 'mesoscopic diffusion'). Next, it discusses transport theories of systems in this limit, and methods of determining their diffusion constant by extracting the velocity autocorrelation of particles in simulations. Finally, it includes results from a molecular dynamics simulation with GROMACS, and the details of preparing and running a force-field dependent simulation on MATLAB. The MATLAB simulation of liquid methyl red (or, otherwise, methyl red in a solvent whose molecules have mass and size properties exactly like itself) gives a value for the diffusion constant to be* $D = (7 \pm 1) \times 10^{-6} \frac{m^2}{s}$*. This is value is significantly different from several experimentally determined diffusion coefficients of methyl red in organic solvents.*

AKNOWLEDGEMENTS

Equally as the development and compilation of this thesis has been rewarding to me, they have demanded the assistance and involvement of several significant individuals. I feel honored to extend my sincere gratitude to Dr. Daniel Spiegel of the Trinity University Department of Physics and Astronomy for his guidance and compassion. I would like to thank the entire Trinity University Department of Physics and Astronomy, as every faculty and staff member has been nothing but patient with and helpful to me. I remain forever indebted to my family, Ma, Papa, Dipto and Bhaiya, and friends for their constant support and encouragement, but most importantly for their presence in my life.

Thank you,

Santona Tuli

I. Introduction

Diffusion lies at the heart of liquid state physics, a prodigious field which physicists often explore using fluid dynamics and statistical mechanics guided by laws of thermodynamics. Models developed in liquid state physics can be used to represent and study many biological and chemical processes. These processes typically involve the interactive dynamics of molecules or species on molecular scales in solution. For instance, consider the biological process of biomixing, the tendency of microscopic organisms (such as bacteria) in a liquid medium to increase the diffusion of other particles scattered in the liquid to increase its chances of encountering useful substances such as food. The subjects of investigation are the processes that enable the organism to achieve this goal. First, however, the rate and mechanisms through which the organism itself as well as its surrounding substances move and diffuse with respect to each other need to be understood. Because all the entities involved, including the bacterium, its surrounding particles and the molecules of the liquid medium, are comparable in size and mass, these entities affect each other's motion through interactions and collisions. The different diffusion mechanisms involved in processes such as biomixing determine the properties of the liquid in question: their understanding is therefore imperative to the prediction and analysis of the fluid's behavior. It is important to identify the relevant diffusion mechanisms or, otherwise, to determine the diffusion mechanism that best describes the fluid motion based on dimensional, structural and interactive parameters of the system. Once the correct diffusion mechanism is known, the diffusion coefficients of the different particle species can be determined experimentally, predicted theoretically, or evaluated from simulations [1, 2] in order to determine the subsequent fluid dynamics and transport. Diffusion coefficients are of interest because their knowledge is vital to the investigation of transport properties of solutions on various length scales.

Classical theories of diffusion on macroscopic scales, which have long been developed using thermodynamics and statistical mechanics, continue to be useful in the study of fluid dynamics [3, 4]. However, a significant amount of research has been dedicated in recent years towards developing transport theories for smaller length scales, such as micro- and nanometer scales [5, 6, 7, 8]. Once developed, these theories will greatly facilitate the study of the dynamical molecular processes needing modeling in the chemical and biological sciences, as the size of relevant molecules in these processes is typically on the order of a few angstroms to a few hundred nanometers. The theories may also help resolve certain discrepancies that arise in diffusivity experiments when the particles involved approach mesoscopic (i.e., nanometer to micrometer) length scales. An example of such a discrepancy is the disagreement between theoretically expected linearity and experimentally observed non-linearity between the diffusion coefficient and the inverse viscosity of certain microscopic solutions [9, 10].

The classical models for macroscopic diffusion become less adequate on the molecular level as microscopic parameters (such as the van der Waal's interaction between particles) which are not considered in these classical models become significant [9, 11]. Research has been geared towards comprehending mesoscopic diffusion and arriving at an analytical expression for the diffusion constant [12]. Meanwhile, an efficient way to determine diffusion constants on the molecular level is through the use of molecular dynamics simulations. In fact, these simulations retroactively assist in the development of transport theories on the mesoscopic scale. Simulations take into account molecular interactions and often employ quantum mechanics in addition to thermodynamics and statistical mechanics [1, 2]. Before the physics behind the simulations can be understood, it is important to understand the existing transport theories. Though the classical model of random walk diffusion is most relevant on macroscopic scales, the theoretical steps of its development are essential in understanding diffusion in general, regardless of the length scales involved. In this thesis I will present a review of the developmental theory of macroscopic diffusion, discuss simulation techniques and their advantages, and finally present some simulation results.

First, let us distinguish the differences in the length scales of diffusion that I have mentioned. A system is macroscopic when the size and mass ratios of solute to solvent molecules are large (for example, for the macroscopic solution of glucose in water, these ratios are, respectively, 3.3 and 10). In this case, the much smaller and lighter solvent molecules have minimal effect on the solute molecules in the dynamical solution. We consider two different categories of effects, the first being the displacement of a solute molecule on the occasion of a collision with a solvent molecule, and the second being the contribution of the solvent molecules to the potential energy of a solute molecule. In macroscopic solutions, the solvent particles lack sufficient mass to significantly distort the potential of the system, and by the same token they lack sufficient momentum to impart a large momentum to the solute molecules upon collision. Because of the comparatively less significant and continuous effect of the solvent molecules on the solute molecules on the macroscopic scale, over long time frames these effects can be statistically averaged and the random walk (or Brownian motion) characteristics of the solute molecules emerge. [3, 4, 5]

The development of classical theories of diffusion, which describe macroscopic rather than mesoscopic systems, involve taking ensemble averages of isolated systems containing single solute particles surrounded by numerous, much smaller solvent particles. The displacements due to the bombardment of the solute molecule with the solvent particles are small enough, and the bombardments occur often enough (compared to time scales used in diffusion measurements), that the net effect is that of continuous *random* displacements of the solute molecule, and over time, the particle's diffusion is proportional to the average of the square displacements. [3, 4, 5] (See detailed discussion of Brownian dynamics in the section II.)

As the size and mass ratios of the solute to solvent molecules approach one or smaller, however, the effects of the interaction and bombardment of the solvent particles on the solute particles grow significant [2, 10, 12]. We call solutions in this limit mesoscopic solutions. If we consider the isolated systems (of a single solute molecule surrounded by numerous solvent molecules) in the mesoscopic scenario, we find that these systems are not statistically similar. That is, now that the solvent molecules are comparable in size and mass to the solute molecules and can therefore affect the solute molecules significantly, no generalizations can be made about ensembles of such isolated systems. For instance, though the solute molecule in some system A may undergo a hundred collisions in a given time frame, it is impossible to claim that some other system B witnesses a similar number of collisions in the same time frame. Furthermore, it also cannot be said that one thousand such systems will undergo a hundred collisions each, on average. Finally, we must also bear in mind that it is not only the number of collisions, but also the nature of the collisions that affect the solute particles trajectory. Therefore, we realize that pure Brownian motion cannot be safely assumed in the case of mesoscopic fluid interactions. [2, 10, 11, 12, 13]

II. Classical Transport Theory:

Brownian Motion for the Macroscopic Scale

On the macroscopic level, i.e. when the size and mass ratios of solute to solvent molecules are large, Brownian motion, or random walk, is responsible for particle diffusion in a homogeneous solution [3, 4, 5]. A rapidly varying *random* force due to the thermal fluctuations in the velocities of the solvent molecules produces the Brownian motion of the solute molecules.

In the absence of external forces, the Brownian motion of a particle of mass, m, in a solvent can be described by the Langevin equation:

$$m\frac{dv}{dt} = -\alpha v + F^{'}(t) \tag{1}$$

where v is the velocity of the particle relative to the solution, α is the friction constant, and $F^{'}(t)$ is a rapidly fluctuating force which arises due to the result of irregular collisions by solvent particles. In thermal equilibrium, the mean displacement of the particle vanishes by symmetry since the particle is equally likely to be displaced in any direction by any given collision. The magnitude of the fluctuations in the displacement therefore gives us a measure of the particle's random diffusion. Next, by multiplying (1) by x we get,

$$mx\frac{d\dot{x}}{dt} = m\left[\frac{d}{dt}(x\dot{x}) - \dot{x}^2\right] = -\alpha x\dot{x} + xF^{'}(t) \tag{2}$$

We can consider the time-evolving system of solute molecules in the solvent as an ensemble of single solute particles surrounded by solvent molecules. Thus, taking ensemble averages on (2), and recognizing that the mean value of the fluctuating force, $F^{'}(t)$, vanishes when we do so, we find (since the operations of taking a time derivative and taking an ensemble average commute) that,

$$m\langle\frac{d}{dt}(x\dot{x})\rangle - m\langle\dot{x}^2\rangle = m\frac{d}{dt}\langle x\dot{x}\rangle - m\langle\dot{x}^2\rangle = -\alpha\langle x\dot{x}\rangle \tag{3}$$

Here, we can use the equipartition theorem result $\frac{1}{2}m\langle\dot{x}^2\rangle = \frac{1}{2}kT$ (where k is the Boltzmann constant and T the absolute temperature) to obtain,

$$m\frac{d}{dt}\langle x\dot{x}\rangle = kT - \alpha\langle x\dot{x}\rangle \tag{4}$$

Equation (4) is a first order ordinary differential equation in the quantity $\langle x\dot{x}\rangle$, which has solutions

$$\langle x\dot{x}\rangle = Ce^{-\frac{t}{\tau}} + \frac{kT}{\alpha} \tag{5}$$

where we have defined the characteristic time constant, τ, by the ratio $\tau = \frac{m}{\alpha}$. We can solve for the constant of integration C using the initial condition $x = 0$ at $t = 0$, to find $C = -\frac{kT}{\alpha}$. Finally, we recall the fact that $\langle x\dot{x}\rangle = \frac{1}{2}\frac{d}{dt}\langle x^2\rangle$ to arrive at the new first order differential equation:

$$\frac{1}{2}\frac{d}{dt}\langle x^2\rangle = \frac{kT}{\alpha}\left(1 - e^{-\frac{t}{\tau}}\right) \tag{6}$$

which we solve for $\langle x^2\rangle$ to get,

$$\langle x^2\rangle = \frac{2kT}{\alpha}\left(t - \tau\left(1 - e^{-\frac{t}{\tau}}\right)\right) \tag{7}$$

In the limiting case that t is much larger than the characteristic time constant, τ, we arrive at the relation,

$$\langle x^2\rangle = \frac{2kT}{\alpha}t \tag{8}$$

which resembles the familiar Brownian motion relation that the mean square displacement is proportional to time.

Using hydrodynamic reasoning to the motion of a macroscopic spherical particle in a liquid and the Navier-Stokes equations to model the flow around the sphere, the friction constant of the solute, α, is defined in the following way by the Stokes law,

$$F = -\alpha v \qquad (9)$$

(where F is the drag force on the particle) and is given by $\alpha = f\pi\eta a$. The variable f represents the degrees of freedom of the solution and is determined by the surface boundary conditions ($f = 4$ or $f = 6$), η is the solvent viscosity and a is the solute particle radius (2). It is important to keep in mind that frictional forces arise in the dynamic description of a system when the surroundings with which the system interacts come to internal equilibrium appreciably quickly compared to the smallest time scale used in the description of the system. That is, our analysis hinges on the fact that we look at specifically the limiting case where $t \gg \tau$.

By comparing equation (8) with the random walk relation $\langle x^2 \rangle = 2Dt$, we arrive at the Stokes-Einstein relation which accurately delineates the classical diffusion coefficient, D:

$$D = \frac{kT}{f\pi\eta a} \qquad (10)$$

Thus, for a macroscopic particle, we are able to determine its diffusion rate in solution by modeling the solution as an ensemble of a large number of systems consisting of just one solute particle surrounded by numerous solvent molecules and using Langevin analysis. The particle-particle potential energy is ignored and the effect of individual solvent molecules on the solute particles is assumed to be sufficiently small as not to produce large changes in displacement of the solute particle upon collision, or, if they do, that these changes average out over long time scales.

[3, 4, 5, 8]

III. The Mesoscopic Scale

However, on the mesoscopic scale, the solvent molecules are comparable in size and mass to the solute molecules and therefore notably affect the solute molecules upon bombardment. Unlike the macroscopic scale, the rapidly varying random force due to the thermal fluctuations in the velocities of the solvent molecules now produces a more complex oscillatory motion of the solute molecules, along with its random walk motion [2, 10]. Therefore, the classical Stokes-Einstein relation breaks down, and experiments with molecular solutions often yield diffusion constants much different from expected results [10, 11, 12]. Due to the lack of a complete transport theory for mesoscopic diffusion, molecular dynamics is often used to simulate the conditions of such solutions, and their results used to determine diffusion coefficients [1, 2, 10, 11, 13]. Though molecular dynamics simulations rely on much of the physics described above, they utilize a force field (or potential) dependent topology for the molecules and rely on correlation functions for calculations of the diffusion constant.

Unlike the case of pure Brownian motion, on the mesoscopic scale, a more detailed investigation of the drag on a spherical particle which is forced to oscillate in a fluid shows that the time dependent drag force $F^{'}(t)$ on a rapidly oscillating sphere requires a non-Markovian generalization of the Langevin equation [14, 15]. Under these conditions, the viscous drag force on the sphere is not simply linearly proportional to the instantaneous velocity of the sphere as in the macroscopic theory described previously (recall Stokes Law, equation (9)). Instead, the viscous drag is linearly proportional to the velocity at all previous times. Thus, the fluctuating drag force for mesoscopic solutions must be evaluated using velocity autocorrelation, which is described in detail in Section a, below. Following this, the origin and nature of the forces on the solute particles in the mesoscopic scale is described in Section b.

a. Velocity Autocorrelation

The velocity autocorrelation function (VAF) is a time dependent correlation function which reveals the nature of the dynamical processes occurring in a molecular system. The VAF is reflective of the forces acting within the system as the nature of the interaction between particles in a system affects the motion of the particles [15, 16]. Consider a system of N atoms, each of whom has some initial velocity, v_i. The VAF is constructed in the following way. At a chosen origin in time, we store all three components of the velocity v_i of each particle (i). That is,

$$v_i(0) = (v_{i,x}(0), v_{i,y}(0), v_{i,z}(0)) \tag{11}$$

We calculate the first contribution to the VAF, corresponding to time zero (i.e. $t = 0$). This is simply the average of the dot products $v_i(0). v_i(0)$ for all atoms:

$$C_v(0) = \frac{1}{N}\sum_{i=1}^{N}\langle v_i(0)\, .\, v_i(0)\rangle \tag{12}$$

At the next time step in the simulation, $t = \Delta t$, and the corresponding velocity for each atom (updated using the force due to the potential) is,

$$v_i(\Delta t) = (v_{i,x}(\Delta t), v_{i,y}(\Delta t), v_{i,z}(\Delta t)) \tag{13}$$

We now calculate the next point of the VAF as,

$$C_v(\Delta t) = \frac{1}{N}\sum_{i=1}^{N}\langle v_i(0) \,.\, v_i(\Delta t)\rangle \tag{14}$$

We can repeat this procedure at each subsequent simulation (or time) step, thereby obtaining a sequence of points of the VAF, as follows:

$$C_v(n\Delta t) = \frac{1}{N}\sum_{i=1}^{N}\langle v_i(0) \,.\, v_i(n\Delta t)\rangle \tag{15}$$

which we may write as,

$$C_v(t) = \langle v(0). v(t)\rangle \tag{16}$$

where the summation is assumed. We stop after a fixed value of n, that is, a fixed number of simulation steps and plot the VAF. We can determine the corresponding diffusion coefficient from the VAF using numerical integration. The relationship is stated in equation (17) below, but first we investigate what the properties of the VAF can tell us about the dynamical system it corresponds to. [16]

Here, we describe three different cases that produce different types of velocity autocorrelation depending on the nature of interaction of particles. First, if the atoms do not interact with each other via a potential, Newton's Laws of motion tell us that the atoms retain their initial velocities for all time, excepted only by the occurrence of random collisions. On the event of these random collisions, the involved atoms receive a new velocity, and then maintain these velocities until another collision, or for all time. Thus, if the atoms do not interact with each other via a potential, all our points in the autocorrelation function $C_v(t) = \langle v(0). v(t)\rangle$ have similar values, and a plot of the velocity autocorrelation function is horizontal on zero after an initially rapid drop from its maximum value at time $t = 0$. Therefore, a VAF plot that is almost horizontal implies that very weak forces are acting in the system. For pure Brownian motion, velocity does not correlate with itself. We have already discussed the statistical analysis employed in the determination of the diffusion coefficient in this case.

Second, if the forces are small but not negligible, the magnitude and direction of the velocities change gradually under the influence of these weak forces. In this case, we expect the dot product $v_i(t_o). v_i(t_o + n\Delta t)$ to decrease on average (for each particle i), as the velocity changes. That is, the velocity decorrelates with time, or, the atoms gradually 'forget' what their initial velocities were. Thus, consider a typical non-horizontal correlation function for random fluctuations in a variable. When it is evaluated at any specific value, $t = t'$, we obtain the maximum amplitude: the mean square value of the variable at that time, which is positive for an autocorrelation function and independent of time. For long time separations, the values of the velocity become uncorrelated. Therefore, $C_v(t)$ acquires its greatest value at $t = 0$, and then tends to zero with time. For randomly fluctuating velocity, with weak net force of particle interaction, the decay of the VAF is exponential (less rapid than in the case of negligible forces), with no oscillations. This indicates the presence of collisions and weak forces slowly destroying the velocity correlation. Such a result is typical of the molecules in a gas, for instance.

Third, we have the case that the interactive forces between the particles are strong. Lennard-Jones potentials provide a good model for such strong interaction of particles in high density systems such as solids and liquids (and solutions) where atoms are packed closely together. Traditionally, Lennard-Jones potentials have been used as the interatomic interaction of molecules in solutions by many research groups simulating solution diffusivity [1, 2, 6, 7, 10, 12]. Under the influence of such a potential, the atoms continually rearrange themselves to reach a state of balance between the repulsive forces and attractive forces acting on them, since this state of balance is the most energetically stable. However, due to their internal energy, and the fact that collisions between particles are a frequent occurrence, the particles cannot stabilize permanently, and therefore constantly move. (This is analogous to Brownian motion, but there are net forces acting on each particle.) In extremely dense systems, like solids, these atomic locations are extremely stable, and the atoms cannot escape easily from their positions. Their motion is therefore oscillatory; the atoms vibrate back and forth, reversing their velocities at the end of each oscillation. If we calculate the VAF of these atoms, we obtain a function that oscillates strongly from positive to negative values and back again. The oscillations will not be of equal magnitude however, but will decay in time, due to the extraneous (fluctuating) forces acting on the atoms which disrupt their oscillatory motion. A plot of such a VAF resembles damped harmonic motion.

Liquids behave similarly to solids, but since the atoms do not have fixed regular positions (they only reach relatively less stable equilibria), they are free to undergo diffusion. The diffusion destroys the oscillatory motion more rapidly than in the case of solids. The VAF in this case is likely to show one damped oscillation (one minimum) before decaying to zero. This may be thought of analogously as a collision between two atoms before they rebound from one another and diffuse away. However, the VAF is dependent on the density of the particles, and more oscillations can be seen for higher density fluids.

As well as revealing the dynamical processes in a system, the VAF can also be used to produce the diffusion coefficient directly. Since the VAF decays to zero at long times (for all of the cases outlined above), the function may be integrated numerically to give the diffusion coefficient D as,

$$D = \frac{1}{3}\int_0^\infty \langle v(0).v(t)\rangle\, dt \qquad (17)$$

This is a special case of a more general relationship between the VAF and the mean square displacement, and is derived from the Green-Kubo relations, which relate correlation functions to transport coefficients.

[15, 16]

b. The Nature of Mescoscopic Diffusion

The net force on the particle in a mesoscopic solution consists of three parts [18, 19, 20]. First, part of the force correlates with the position: this is the force due to potential arising from the force field (particle-particle interactions). A second part of the force is frictional, and therefore correlates with velocity, as explained in section a. Finally, there exists a third part which is neither correlated with velocity nor position. This can be thought of as 'noise' and its source is the same sort of randomness or fluctuation which explicates random walk. The noise is characterized by stochastic properties, the most important of which is that it is self-correlated. The self-correlation function is expressed as $\langle R(0)R(t)\rangle$, where R is the noise component of the force on the particle. Thus, though on a macroscopic scale the fluctuations which we call noise become negligible and the Navier-Stokes equations of continuum fluid dynamics emerge, on a mesoscopic scale, these fluctuations remain important and Stokes Law is no longer adequate to define the friction constant α we encountered before. [19, 20]

In both the macro- and mesoscopic scales, the transport properties described are properties of systems at equilibrium. Both the Langevin equation for the irregular Brownian motion of macroscopic particles, and the generalized non-Markovian Langevin for the oscillatory motion of a mesoscopic particle describe systems at equilibrium. Recall that the Stokes Law for the viscous drag on a sphere which is forced to move with constant velocity through a fluid tells us that the friction coefficient $\alpha = f\pi\eta a$. The Onsager regression hypothesis suggests that this implies that equilibrium fluctuations can be modeled by non-equilibrium transport coefficients, in this case the shear viscosity, η, of the fluid. Therefore, the inverse of this hypothesis is also true: knowledge of the equilibrium fluctuations allows us to calculate transport coefficients. [3, 4, 5, 19]

Keeping the three components of the force on a solute particle in a mesoscopic solution in mind, we propose the following. When taken over time scales which are large compared to the noise correlation time, the average total energy of the system must be conserved. This is because the systematic force arising due to the particle-particle potential is conservative. Now, since frictional force itself is dissipative and decreases kinetic energy, and the stochastic (or fluctuating) force in first order does not affect the total energy but in second order increases the kinetic energy, the cooling due to friction must cancel the heating by noise. This relationship between friction and noise is known as the fluctuation-dissipation theorem [17, 18, 20, 21]. Due to the fluctuation-dissipation theorem, we can use the noise self-correlation to determine the friction constant (instead of the Stokes-Einstein relation in the macroscopic scale), which we can then use to determine the diffusion constant. Using Green-Kubo formulas we arrive at the relation

$$D = \frac{(kT)^2}{\int_0^\infty \langle R(0)R(t)\rangle dt} \qquad (18)$$

From our analysis of the fluctuation-dissipation theorem, this noise autocorrelation suggests that the velocity is also autocorrelated with time. The exact relationship between the velocity autocorrelation function (VAF), $C_v = \langle v(0).v(t)\rangle$, and the diffusion coefficient is given by, equation (17) (above). [3, 4, 5]

Equation (17) is used to determine the diffusion coefficient in the all the simulations described below. In the simulations, a velocity autocorrelation function is generated by determining the velocity correlation at each simulation step. The VAF is then numerically integrated and the result divided by 3 to obtain the diffusion coefficient, D.

IV. Simulations and their Results

a. Discussion of GROMACS Simulation

In Fall 2012, I used the molecular dynamics engine GROMACS (version 4.5.5) to simulate a solution of phenylalanine in methanol [24, 25]. GROMACS is a molecular dynamics simulation engine popularly used for building complex biological simulations (for example, protein-lipid systems). Molecular dynamics engines such as GROMACS can output diffusion constants of the particles involved based on statistical correlation of particle trajectory, velocity, acceleration etc. Below is a discussion of the simulation techniques used by GROMACS and a discussion of my results. Appendix A contains the details of the commands required to run the simulation. [21]

The temperature, pressure and other deterministic parameters can be specified for the particular solution we want to simulate, and the GROMACS topologies (or protein data bank (pdb) files in the correct format) for the solute and solvent molecules have to be provided. Topologies take into account the particle-particle interactions and set up the relevant potential. The force field used to determine the particle-particle interactions must therefore be specified. The simulated solution is then allowed to progress in time. Simulation 'frames' yield mechanical observables, like atomic distances, velocities, energies, pressure etc. which are immediately measurable in each configuration. By averaging over an ensemble of configurations, particle trajectories and certain thermodynamic properties like distribution functions and energy, temperature and pressure profiles are obtained [19, 20, 21, 22]. Finally, by observing the time dependence of equilibrium fluctuations, transport properties such as diffusion constant and viscosity can be calculated from the simulations (using the inverse of the Onsager regression hypothesis). [4, 5, 18, 21]

The largest hindrance in using simulations of molecular dynamics to model the required diffusion conditions with atomic details is the available computational power and consequent limit to time scales over which the simulations can be designed [19, 20, 21, 22, 23]. (Currently, these time scales are in the order of 100 ns.) Thus, for simulations that are required to be in the micro- or millisecond range, as well as system sizes beyond 100,000 particles, it is necessary to find methods to simplify the system. The key is to reduce the number of degrees of freedom. Once we define important degrees of freedom, we can average out the 'unimportant' degrees of freedom in such a way that the thermodynamic and long time-scale properties are preserved. [18, 21]

The technique for reduction of degrees of freedom depends on the particularities of the system to be simulated. One approach is the use of superatoms (or pseudoparticles), which is the term used to describe the lumping together of several atoms into one interaction unit. The interactions change into potentials of mean force, and the omitted degrees of freedom are replaced by noise and friction. The force due to quantum degrees of freedom on classical degrees of freedom determines the classical dynamics and thus the time-dependence of the field in which the quantum system evolves. The interaction units we use in our GROMACS simulation are the phenylalanine and methanol molecules. We chose Lennard-Jones potentials as the interaction between superatoms, and using the relevant force field we produce the topologies of the molecules. [20, 21]

By running the GROMACS simulation of phenylalanine in methanol at a temperature of 500 K we determine $D = 2.81 \times 10^{-10} m^2/s$. The order of magnitude suggests that this result may be reasonable [26, 27]; however, we have reasons to suspect this result is not valid. The temperature for this simulation was chosen somewhat arbitrarily, and a more suitable temperature for the simulation may be room temperature (300 K) in order for the results to be comparable to the data from laboratory experiments. In fact, since methanol boils at 340 K, the system we simulated represents a physically unrealistic ‘solution’. There also remains some uncertainty regarding the charge balance of the simulation box which could invalidate our results. Since both phenylalanine and methanol are neutral, organic compounds, it is reasonable to expect the mixture to be neutral. However, if the solution is not neutral, the excess charges must be balanced before running the molecular dynamics in order for the model for our long range electrostatic interactions to be valid. [21, 24, 25]

Finally, the actual chemical solution we were interested in in laboratory experiments is methyl red in butanol [28]. In Spring 2013, I attempted to set up a simulation of this solution using GROMACS, but ran into some difficulties along the way. The commands I had run before to conduct the GROMACS simulation could not be executed on the new pre-configured computational system on which GROMACS was installed. When the system eventually crashed, I decided to create alternative MATLAB simulations for molecular dynamics involving Lennard-Jones potential. MATLAB simulations are more operationally transparent and allow much more user control. Their disadvantage is that they are incapable of incorporating the structure of molecules, and as a result treat the molecules only as point particles.

b. Discussion of MATLAB Simulation

The MATLAB code for a simulation using Lennard-Jones potentials for particle interaction representing mesoscopic diffusion can be found in Appendix B [23]. What follows is a description of the design of the code. A three-dimensional cubic simulation box consisting of 125 identical particles is defined. Here, we consider these particles to be pseudoparticles representing molecules. The particles' masses are scaled by their actual mass in kilograms. Thus, suppose we are simulating a small part of a sample of liquid methyl red (mass per molecule = 4.47×10^{-25} kg), then the mass of the simulation particles of 1 mass unit is equivalent to this kilogram value. Instead of defining the dimensions of the box, we say the density of the particles is 0.85 mass units/unit volume. The actual density of methyl red is 0.791 g/cm^3 or 1.77×10^{21} molecules/cm^3. Therefore, the volume occupied by each molecule in our simulation is $(0.85 \times 4.47 \times 10^{-25})/(1.77 \times 10^{21})$ cm^3. The total volume occupied by all 125 molecules (assuming they are close to each other) is $125 \times 4.81 \times 10^{-22}$ cm^3 = 6.00×10^{-20} cm^3, and the length of the sides of our box is 3.92×10^{-7}cm (see function initialCubicConfiguration in Appendix B).

We use integration time steps of 0.0001 seconds and 10,000 integration steps, which give us a total simulation time of 1 second. Since the characteristic time, $\tau = \frac{m}{\alpha}$, is on the order of pico- to micro-seconds, a simulation time of 1 second is sufficiently large to allow us to use the methods outlined above to determine diffusion coefficients. We keep track of the progress of the simulation by printing the number of integration steps performed in increments of 100 integration steps.

To generate the initial coordinates of all the particles, the code calls the function called initialCubicConfiguration (Appendix B), which ensures that the particles are distributed precisely in a cubic lattice. Next, three dimensional initial velocities for all the particles are randomly generated using the function randomGaussian (Appendix B). The velocities are generated about a mean of zero and a standard deviation, $velsSTD = \sqrt{\frac{Temp}{mass}}$, which represents the statistical standard deviation of velocity from the mean velocity of an ensemble of massive particles in a heat bath (a thermostat parameter). We implement an Andersen thermostat for the heat bath defining a frequency of collisions of particles with the heat bath. $Temp$ is the simulation temperature, scaled by the Boltzmann constant times room temperature and $mass$ is the scaled mass (as above). Note that, therefore $Temp = 2$ corresponds to a kT factor of $kT = 2 \times kT_{room} = 8.28 \times 10^{-21}$ J. Thus, the scaling factor for $velsSTD$ is

$\sqrt{8.28 \times 10^{-21}\,J \big/ 4.47 \times 10^{-25} kg}^{\,-1} = (\sqrt{18523}\,m/s)^{-1}$. Since we generated the velocities using $velsSTD$, the velocities are also scaled by this factor.

We then find the center of mass velocity of the particles, and subtract off this value from the particle velocities in order to eliminate center of mass drift. The next step is to set the initial kinetic energy to the internal energy of the system. We find the mean squared velocity of the particles by finding the norm squared of the (initial) velocity vector and dividing by the number of particles. Next we define the velocity scaling factor as the square root of the product of the number of dimensions, the temperature and the inverse mean squared velocity. This is essentially a proxy for the Boltzmann constant. We then scale the velocity in each dimension for each particle by the scaling factor. This is the appropriately scaled initial velocity matrix for the particles and we store it as velsInitial to use in determining the initial velocity autocorrelation. The method for finding the velocity autocorrelation is outlined in Section a Part III, above. We take the dot product of the initial velocities (in each dimension) to find the norm squared. Next, we sum the columns of this matrix over all the particles and then divide by the number of particles in order to obtain the initial velocity correlation in each dimension.

The next step is to calculate the initial forces on the particles. We do so using the function LennardJones_Force (Appendix B). For all particle pairs, we determine their separation and apply to it the periodic boundary conditions of the simulation box by calling the function PBC_3D (Appendix B), which simply adjusts the distance between any two points using the half-length of the simulation box (in three dimensions). Next, the LennardJones_Force function uses the inverse square of the inter-particle separation for each particle pair, to find the appropriate Lennard-Jones force factor between the two particles. The Lennard-Jones potential is given by

$$U(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{19}$$

where r is the separation between the two particles, σ is the finite distance at which the inter-particle potential is zero and ε is the depth of the potential well. We scale our parameters by these distance and energy units by setting $\sigma = 1$ and $\varepsilon = 1$. This allows us to use the Lennard-Jones potential to describe the forces in the model without initial knowledge of the specific details of the depth and width of the potential well. As more simulations are run in the future gearing towards better accuracy of results, these parameters can be assigned actual values whose knowledge can be derived from previous simulations or from viscosity calculations. Then the magnitude of the attractive force in each of the three dimensions, found by taking the derivative of the Lennard-Jones potential is given by

$$F(r) = 24\left[2\left(\frac{1}{r}\right)^{13} - \left(\frac{1}{r}\right)^{7}\right] \tag{20}$$

We find the force on each particle due to all the other particles by looping over all particle pairs and multiplying this force factor by the separation distance and summing over these values.

We are now ready to start the molecular dynamics. During the first integration step, we update the positions of all the particles using the equation of motion:

$$s = s_o + ut + \frac{1}{2}at^2 \tag{21}$$

where s_o is the initial position of a particle (‘coords’ in the code), u is its instantaneous velocity (‘vels’ in the code), a is its acceleration (for which ‘forces’ in the codes acts as a proxy, since it is scaled by the mass), and we use the integration time ‘dt’ as the time elapsed. After applying the periodic boundary conditions on the coordinates again and calculating the new forces on the particles using their new positions, we are ready for the second integration step.

We update the velocities using the new forces, implementing the Andersen’s thermostat to ensure that particles which collide with the heat bath receive a new randomly generated velocity. This concludes the relevant calculations for each integration step in the simulation. We find the correlation function for each integration step by finding the velocity norm squared in each dimension for each particle and finding the average of these values for all the particles. We record the velocity autocorrelation, and then move time forward. Looping these steps over all the 10,000 integration steps allows us to conduct our simulation. Within this loop we also sample the velocities of the particles in the x direction in order to keep track of the nature of their distribution. This simulation takes roughly two hours and thirty minutes to complete running.

The final part of the MATLAB code allows us to generate the results. The simulation outputs a plot of the velocity autocorrelation function. We ran the simulation five times to obtain the following results. Figure 1 shows the VAF from one of the runs; its corresponding velocity histogram and diffusion coefficient are discussed below.

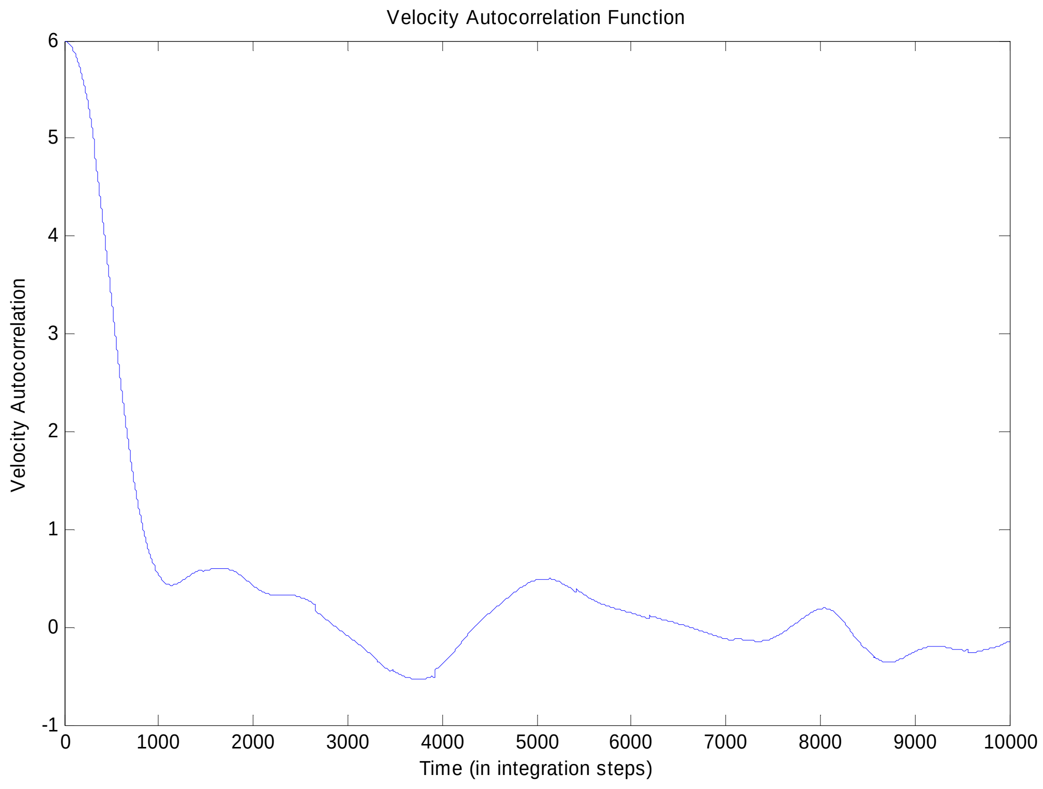

Figure 1: Velocity Autocorrelation Function for a Simulation Run

The VAF resembles what we described as the third kind of the velocity autocorrelation function in Section a Part III. Since we used Lennard-Jones potentials to model the interaction of the particles in order to simulate a solution, the interactive forces between the particles are relatively strong. The oscillatory nature of the motion of the particles is represented by the oscillations of the VAF about zero. The oscillations decay gradually, due to the fluctuating forces acting on the atoms which disturb their oscillatory motion. In other words, since the particles in solution diffuse, the oscillatory motion of the particles and the oscillations of the VAF are gradually destroyed. Nonetheless, we can tell that the VAF belongs to a high density solution because we can see several oscillations in the VAF.

In addition, we sample the x components of the velocities of the particles for the same simulation run and record them in the following histogram (Figure 2).

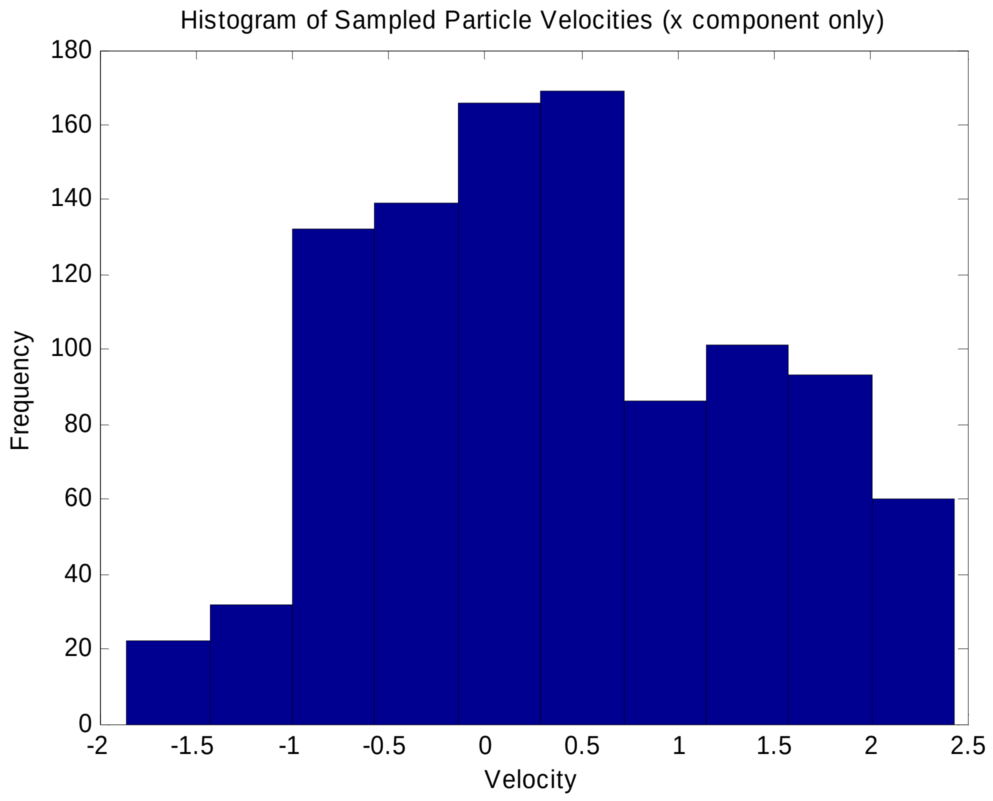

Figure 2: Histogram of Velocities (x Component) for the same Simulation Run

The velocities appear to be distributed over a range centered roughly about zero. By averaging over all the simulation runs, we notice that the (x components of the) velocities are distributed roughly normally about zero, which is what we expect since we generate our initial velocities using a Gaussian distribution and because of the random nature of the fluctuating forces on which we base our predictions and analyses (recall the fluctuation-dissipation theorem).

Next, we use the velocity autocorrelation functions generated by our simulations in order to find the diffusion coefficient of the solution as described in Section a and b Part III. In MATLAB, we use the command *trapz* to perform the numerical integration of the VAF. The VAF shown in Figure 1 and the velocity histogram shown in Figure 2 correspond to a diffusion coefficient of $D = 1.26 \times 10^3$ in scaled units. Now we perform a quick rescaling to obtain the diffusion coefficient in S.I. units. From equation (17) we know that D is proportional to the product of $(vels)^2$ and time (in integration steps). Thus, the scaling factor for D is $\{(\sqrt{18523}\, m/s)^{-1}\}^2 \times 10000 s^{-1} = (1.85 \times 10^8\, m^2/s)^{-1}$. Therefore, the physical value of the diffusion coefficient from the simulation is $D = \frac{1.26\times10^3}{1.85\times10^8} \frac{m^2}{s} = 6.80 \times 10^{-6} \frac{m^2}{s}$. Table 1 below lists the diffusion coefficients obtained from running the MATLAB simulation five times after converting them to physical units.

Run Number	1	2	3	4	5
Diffusion Coefficient in $\frac{m^2}{s}$	8.54×10^{-6}	5.32×10^{-6}	7.73×10^{-6}	6.00×10^{-6}	6.80×10^{-6}

Table 1: Table of Diffusion Coefficients from Simulations.

The average value for the diffusion coefficient obtained from these results is $D = 6.88 \times 10^{-6} \frac{m^2}{s}$. The standard deviation of these diffusion coefficients is $1.29 \times 10^{-6} \frac{m^2}{s}$, and therefore, in standard form, $D = (7 \pm 1) \times 10^{-6} \frac{m^2}{s}$. Previously documented values for the thermal diffusion coefficient of methyl red in experiments involving transient gratings include the following: $D = 1.1 \times 10^{-7} \frac{m^2}{s}$ in benzene, $D = 9.33 \times 10^{-8} \frac{m^2}{s}$ in ethanol and $D = 7.81 \times 10^{-8} \frac{m^2}{s}$ in 2-propanol [25], and $D = 2.10 \times 10^{-9} \frac{m^2}{s}$ in 2-propanol [26]. The value for the diffusion coefficient of methyl red from our simulation is roughly seven standard deviations away from each of these values. Our simulation is used to determine the self-diffusion rate of methyl red. The above mentioned solvent molecules are smaller and less massive than methyl red molecules. Moreover, in the simulation, the methyl red molecules are closely packed together, more so than they would be in a solution in another solvent. Thus, it is reasonable that the diffusion coefficient we found using simulations is higher than the experimentally determined diffusion coefficients in previous literature. Furthermore, the results of our simulations can be enhanced by greater knowledge of the specific details of the Lennard-Jones potential between methyl red molecules.

V. Conclusion

Though the classical Stokes-Einstein relation appropriately describes diffusion on a macroscopic scale, on the mesoscopic scale we rely on the fluctuation-dissipation theorem and the autocorrelation of the equilibrium fluctuations to calculate diffusion constants. Molecular dynamics simulations are extremely useful tools for simulating solutions with atomic details and evaluating their diffusion constants using Green-Kubo formulae. Using the molecular dynamics engine GROMACS (version 4.5.5), we determine the diffusion constant of a phenylalanine in methanol solution at 500 K to be $2.81 \times 10^{-10} m^2/s$. We also write a script for molecular dynamics simulations on MATLAB. This provides a more user friendly interface and allows greater user control over the parameters and potentials to be used. From the MATLAB simulations, we determine the diffusion coefficient of a solution of $D = (7 \pm 1) \times 10^{-6} \frac{m^2}{s}$, which is significantly different from values of the diffusion coefficient for methyl red in various organic solvents that appear in the literature. Nonetheless, the details of our simulation lead us to believe that the value for the diffusion coefficient we have obtained is reasonable for the specific condition we are simulating.

The primary purpose of my research and documentation has been to make simulation tools readily available for associated research work. Forced Rayleigh Scattering experiments in the laboratory can be used to make diffusion coefficient measurements over a temperature range [24] in order to compare with results produced from simulations as well as expected results from theory. Moreover, projects on biomixing can greatly facilitate from simulations that approximate mesoscopic diffusion. Finally, these simulations may also be used in the investigation of the relationship of viscosity to diffusion. An easy way to vary the viscosity of the simulation in simulations is to vary the temperature of the solution. Since the relationship between viscosity and diffusion coefficients diverges from expectations from classical transport theories, these simulations can promise an efficient way to investigate this divergence and to attempt to explicate them. Though some of these projects are more practically oriented than others, they all ultimately converge towards the goal of improving our understanding of mesoscopic diffusion. Research in this field promises the exciting prospect of developing transport theories for molecular solutions and proposing an analytic expression for the diffusion coefficient on this length scale.

References:

1. McPhie, Mathieu G., Daivis Peter J., and Snook Ian K., Phys. Rev. E 74, 031201 (2006)
2. S. K. Kumar, G. Szamel, and J. F. Douglas, J. Chem. Phys. 124, 214501 (2006)
3. Reed, Thomas M. and Gubbins, Keith E., *Applied Statistical Mechanics* (Butterworth-Heinemann, Stoneham, 1973).
4. Chandler, David, *Introduction to Modern Statistical Mechanics*(Oxford University Press, Oxford, 1987)
5. Eyring, Henry, Henderson, Douglas, Stover, Betsy J. and Eyring, Edward M., *Statistical Mechanics and Dynamics* (Wiley, New York, 1982)
6. R. I. Cukier and J. M. Deutch, Phys. Rev. 177, 240 (1969)
7. Kravchenko, Olga and Thachuk, Mark, J. Chem. Phys. **134**, 114310 (2011); http://dx.doi.org/10.1063/1.3562369
8. Rah, Kyunil and Eu, Byung Chan, J. Chem. Phys. 116, 7967 (2002); doi: 10.1063/1.1468218
9. Sarman S. and Evans D.J., J. Chem. Phys. 99, 9021 (1993).
10. F. Ould-Kaddour and D. Levesque, J. Chem. Phys. 127, 154514 (2007); doi: 10.1063/1.2794753
11. I. W. Hamley, *Introduction to Soft Matter* (Wiley, New York, 2002) doi:10.1063/1.1678669.
12. S. H. Lee and R. Kapral, J. Chem. Phys. 121, 11163 (2004).
13. Z. Li, Phys. Rev. E 80, 061204 (2009)
14. F. Ould-Kaddour and D. Levesque, Phys. Rev. E 63, 011205 (2000)
15. Hgnggi, P., Z. Physik B 31, 407-416 (1978)

16. Democritus. Website: The Velocity Autocorrelation Function. http://www.compsoc.man.ac.uk/~lucky/Democritus/Theory/vaf.html

17. MIT lecture series: 22.103 Microscopic Theory of Transport (Fall 2003). Lecture 3: Diffusion and the Velocity Autocorrelation – Green-Kubo Relations. Dec 2003.

18. R. M. Mazo, Brownian Motion, Fluctuation, Dynamics, and Applications (Clarendon, Oxford, 2002)

19. Cai, Wei. Handout: An Overview of Molecular Simulation. September 2005.

20. H. J.C. Berendsen. Lecture: Beyond Molecular Dynamics. Feb 2004.

21. J. E. Kerrigan. Gromacs Introductory Tutorial.

22. Keffer, David. Lecture: The Working Person's Guide to Molecular Dynamics Simulations. 2002.

23. Benjamini, Ayelet. Website: Understanding Molecular Simulations. March 2012.

24. Tuli, Santona, *Diffusion Analysis of Mesoscopic Solutions using Molecular Dynamics,* Fall 2012 Research Report, Trinity University.

25. Tuli, Santona. Presentation: Molecular Dynamics. Fall 2012.

26. Terazima , Masahide, Okamoto, Koichi, and Hirota, Noboru, *J. Phys. Chem.* 97, 5188 (1993)

27. Tuli, Santona, *Using Transient Gratings to Calculate Diffusion Coefficients,* Fall 2010 Research Report, Trinity University.

28. Tuli, Santona, *Investigating the Effects of Hydrogen Bonding on Diffusion Rates in Cis-Trans Isomers of Methyl Red,* Spring 2012 Research Report, Trinity University.

Appendix A

Commands for running GROMACS:

After installation of GROMACS (version 4.5.5) with fftw support (fftw version 2.3.3), we obtain the protein data bank (pdb) files for the solute and solvent molecules. These can be obtained from the protein data bank website, but they need to be edited to the format required by GROMACS. Fortunately the phenylalanine pdb file is available in the GROMACS package and we use this directly. We use the command

pdb2gmx –ignh –ff gromos43a1 **–f** phenylalanine.pdb **–o** phenylalanine.pdb **–p** phenylalanine.top **–water** space

to process the phenylalanine pdb file. The **–ff** flag selects the force field (gromos43a1, an united atom force field), **–f** takes the input pdb, **–o** and **–p** produce the appropriate output files, and **water** lets us specify the water environment of the solution. In this manner, we obtain the topology for phenylalanine. The methanol topology file preexists within GROMACS as well and we use this directly as well. Now we are ready to set up the simulation box.

The command:

editconf -bt octahedron **–f** phenylalanine.pdb **–o** phenylalanine-b4sol.pdb **–d** 4.0

lets us specify the details of the box. The flag **–d** 4.0 tells GROMACS to use sides of length 4 nm. Next, we solvate the box using

genbox –cp phenylalanine-b4sol.pdb **–cs** methanol.gro **–o** phenylalanine-b4ion.pdb **–p** phenylalanine.top where the **–cs** flag allows us to select the solvent. (The command **editconf –f** methanol.pdb **–o** methanol.gro is used to convert the methanol pdb file to a file format suitable for using to solvate the box.) We are now ready to set up the run by performing energy minimization. We pre-process the energy minimization using,

grompp –f em.mdp **–c** phenylalanine-b4ion.pdb **–p** phenylalanine.top **–o** ion.tpr

where the em.mdp file lets us specify various parameters such as:

- constraints – sets any constraints used in the model.
- integrator (steep) – tells grompp that this run is a steepest descents minimization.
- dt – needed for molecular dynamics integrators.
- nsteps – the maximum number of iterations in minimization runs.
- nstlist – frequency to update the neighbor list. We use nstlist = 10.
- rlist – cut-off distance for short-range neighbor list.
- coulombtype – tells gromacs how to model electrostatics. We use PME.
- rcoulomb – distance for the coulomb cut-off.
- vdwtype – tells Gromacs how to treat van der Waals interactions. We use cut-off.
- rvdw – distance for the LJ potential cut-off.
- fourierspacing – Used to automate setup of the grid dimensions for PME.
- emtol – the minimization converges when the max force is smaller than this value (in units of kJ mol^{-1} nm^{-1})
- emstep – initial step size (in nm) for minimization.

Next, we run the energy minimization using,

mdrun –v –deffnm em

We can monitor the progress of the minimization using the **tail** command to check on the progress of the minimization:

```
tail –15 em.log
```

Similarly, we pre-process and then run the pr.mdp (position-restrained) and md.mdp (molecular dynamics) files after editing them with the appropriate parameter values for our simulation. The run commands are

```
nohup mdrun –deffnm pr &
```

and

```
nohup mdrun –deffnm md &
```

respectively. The last statement is the true simulation run statement. The phenylalanine in methanol simulation took about 27 minutes. Though these steps did not involve calling an MPI engine explicitly, GROMACS may have used MPI support by calling an MPI engine within the software or on the computer. The simulation is likely to have used 1 core, but may have used 4 cores instead.

We are now ready for analysis. To calculate the diffusion constant of the particle via the Green-Kubo equation we use the **g_velacc** command**.** This gives us the file 'vac.xvg' with a linear velocity autocorrelation function (VAF). Next, to obtain the diffusion constant we compute the integral of the VAF using:

```
g_analyze -f vac.xvg -integrate
```

The integral is equal to the diffusion constant in units of nm^2/ps.

If the solution is not neutral, the excess charges must be balanced before running the molecular dynamics in order for the model for our long range electrostatic interactions to be valid. For example, the command to add two chloride ions in order to balance two excessive positive charges is as follows:

genion –s ion.tpr **–o** fws-b4em.pdb **–nname** CL **–nn** 2 **–p** fws.top **–g** ion.log

Appendix B

Primary MATLAB code:

```
clear all; close all;

% Determining the Diffusion Coefficient of Particles in a
Lenard-Jones
% Potential using Molecular Dynamics Simulation.

% Written by Santona Tuli, Spring 2013.


    % ===================
    %     Initialize
    % ===================

    % Set simulation box parameters
    nPart = 125;        % Number of particles
    density = 0.85;     % Density of particles
    mass = 1;           % Particles' mass
    nDim = 3;           % The dimensionality of the system (3D
in our case)

    % Set simulation parameters
    dt = 0.0001;        % Integration time
    dt2 = dt*dt;        % Integration time, squared
    Temp = 2.0;         % Simulation temperature
    nu = 0.1;           % Thermostat parameter - frequency of
collisions with the heat bath
    velSTD = sqrt(Temp/mass); % Thermostat parameter - standard
deviation of the velocity
```

```
    nSteps = 10000;     % Total simulation time (in integration
steps)
    sampleFreq = 10;    % Sampling frequency
    sampleCounter = 0;  % Sampling counter
    printFreq = 100;    % Printing frequency

    % Initialize matrices used for correlation
    correlation = zeros(nSteps+1,1); % The Velocity
Autocorrelation as a function of time, for the complete
simulation
    % The following are only for understanding their sizes
    % velsInitial = zeros(nDim,nPart);
    % velsInitialDot = zeros(1,nPart);
    % sumInitial = 0;
    % velsdtDot = zeros(nSteps,nPart);
    % sumdt = zeros(nSteps,1);

    % Set initial configuration
    [coords L] = initialCubicConfiguration(nPart,density,mass);
% See fuction initialCubicConfiguration

    % Set initial velocities with random numbers
    vels = zeros(nDim,nPart);
    for part=1:nPart
        vels(:,part) = randomGaussian(0,velSTD,nDim);  % See
fuction randomGaussian
    end

    % Set initial momentum to zero
    totV = sum(vels,2)/nPart;  % Center of mass velocity
    for dim=1:nDim
        vels(dim,:) = vels(dim,:) - totV(dim);   % Fix any
center-of-mass drift
    end

    % Set initial kinetic energy to nDim*KbT/2
    totV2 = sum(dot(vels,vels))/nPart;  % Mean-squared velocity
    velScale = sqrt(nDim*Temp/totV2);   % Velocity scaling
factor
    for dim=1:nDim
```

```
        vels(dim,:) = vels(dim,:)*velScale;
    end

    % Store the initial velocities for corrrelation
    velsInitial = vels;

    % Find the initial contribution to the VAF (velocity
autocorrelation
    %function)
    % Dot product of the x, y, z components of the velocity of
each particle
    velsInitialDot = dot(velsInitial,velsInitial,2);

    % Sum of the velocity dot products over all the particles
    sumInitial = sum(velsInitialDot);

    % The initial correlation (average over all the particles)
    cInitial = (1/nPart)*sumInitial;

    correlation(1) = cInitial;


    % Calculate initial forces
    forces = LenardJones_Force(coords,L);  % See function
LenardJones_Force



    % ===================
    % Molecular Dynamics
    % ===================

    time = 0; % Following simulation time

    for step = 1:nSteps

        % === First integration step ===

        % Update positions - All coordinates are updated at once
        coords = coords + dt*vels + 0.5*dt2*forces;
        % Apply periodic boundary conditions
        for part=1:nPart
            coords(:,part) = PBC_3D(coords(:,part),L);  % See
function PBC_3D
        end

        % Update velocities - All velocities are updated at once
        vels = vels + 0.5*dt*forces;

        % === Calculate new forces ===
        forces = LenardJones_Force(coords,L);

        % === Second integration step ===

        % Update velocities - All velocities are updated at once
        vels = vels + 0.5*dt*forces;

        % Implement the Andersen thermostat
        for part =1:nPart
            % Test for collisions with the Andersen heat bath
            if (rand < nu*dt)
                % If the particle collided with the bath, draw a
new velocity
                % out of a normal distribution
                vels(:,part) = randomGaussian(0,velSTD,nDim);
            end
        end

        % === Move time forward ===
        time = time + dt;

        % === Find contribution to the VAF ===
        velsdtDot = dot(velsInitial,vels,2);

        sumdt = sum(transpose(velsdtDot));

        cdt = (1/nPart)*sumdt;  % Correlation at time dt, or at
each simulation step
```

```
        correlation(step+1) = cdt;  % Velocity Correlation
Function for each particle for entire simulation

        % === Sample ===
        if mod(step,sampleFreq) == 0
            sampleCounter = sampleCounter + 1;
            % Sample x velocities for display
            for part=1:nPart
                v(sampleCounter) = vels(1, part); % Put all
sampled velocities in a new vector (only x)
            end
        end

        if mod(step,printFreq) == 0
            step % Print the step
        end

    end



    % ===================
    % Simulation results
    % ===================

    % Velocity Autocorrelation
    VAF = plot(correlation);  %  Plots the x, y and z velocity
correlation functions against the simulation steps
    title('Velocity Autocorrelation Function');
    xlabel('Time (in integration steps)');
    ylabel('Velocity Autocorrelation');
    saveas(VAF,'VAFn.fig')

    % Display histogram of sampled x velocities
    figure
    hist(v)  % Draw the histogram of the sampled velocities (x
only)
    title('Histogram of Sampled Particle Velocities (x component
only)');
    xlabel('Velocity');
    ylabel('Frequency');

    % Diffusion Coefficient
    D = (1/3)*trapz(correlation)  %Numerical integration
```

Function: initialCubicConfiguration

```
function [coords, L] =
initialCubicConfiguration(nPart,density,mass)

        % Initialize with zeroes
        coords = zeros(3,nPart);

        % Get the cooresponding box size
        L = ((nPart*mass)/density)^(1.0/3);

        % Find the lowest perfect cube greater than or equal to
the number of
        % particles
        nCube = 2;

        while (nCube^3 < nPart)
            nCube = nCube + 1;
        end


        % Start positioning using a 3D index for counting the
spots
        index = [0,0,0]';

        % Assign particle positions
        for part=1:nPart
            % Set coordinate
            coords(:,part) = (index+[0.5,0.5,0.5]')*(L/nCube);

            % Advance the index
            index(1) = index(1) + 1;
            if (index(1) == nCube)
                index(1) = 0;
                index(2) = index(2) + 1;
                if (index(2) == nCube)
                    index(2) = 0;
                    index(3) = index(3) + 1;
                end
            end
        end

    end
```

Function: randomGaussian

```
function randNums = randomGaussian(mu,sigma,nDim)

        % Generate normally distributed random numbers
        randNums = randn(nDim,1);

        % Shift to match given mean and std
        randNums = mu + randNums * sigma;

    end
```

Function: LenardJones_Force

```
function forces = LenardJones_Force(coords,L)

        % Initialize all forces to 0
        forces = zeros(size(coords));

        % Get the number of particles
        nPart = size(coords,2);

        % Loop over all particle pairs
        for partA = 1:nPart-1
            for partB = (partA+1):nPart

                % Calculate particle-particle distance
                dr = coords(:,partA) - coords(:,partB);
                % Fix according to periodic boundary conditions
                dr = PBC_3D(dr,L);
                % Get the distance squared
                dr2 = dot(dr,dr);

                % Lennard-Jones potential:
                % U(r) = 4*epsilon* [(sigma/r)^12 - (sigma/r)^6]
                %
                % Here, we set sigma = 1, epsilon = 1 (reduced
distance and
                % energy units). Therefore:
                %
                % U(r) = 4 * [(1/r)^12 - (1/r)^6]
                %
                % Fx(r) = 4 * (x/r) * [12*(1/r)^13 - 6*(1/r)^7]
                %
                %       = 48 * x * (1/r)^8 * [(1/r)^6 - 0.5]
                %
                % And same goes for the force in the y&z
directions.
                %
                % For efficiency, we will multiply by 48 only
after summing
                % up all the forces.

                invDr2 = 1.0/dr2; % 1/r^2
                forceFact = invDr2^4 * (invDr2^3 - 0.5);

                % According to Newton's third law, we get action
and
                % reaction for the two particles.
                forces(:,partA) = forces(:,partA) +
dr*forceFact;
                forces(:,partB) = forces(:,partB) -
dr*forceFact;

            end
        end

        % Multiply all forces by 48
        forces = forces*48;

    end
```

Function: PBC_3D

```
function vec = PBC_3D(vec,L)

    % Calculate the half box size in each direction
    hL = L/2.0;

    % Distance vector should be in the range -hLx -> hLx, -hLy -
> hLy and
    % -hLz -> hLz
    % Therefore, we need to apply the following changes if it's
not in this range:

    for dim=1:3
        if (vec(dim) > hL)
            vec(dim) = vec(dim)-L;
        elseif (vec(dim) < -hL)
```

```
            vec(dim) = vec(dim)+L;
        end
    end


    end
```