SIMULATION OF THE FLOW OF A SINGLE STRANDED DNA IN A CHANNEL USING DISSIPATIVE PARTICLE DYNAMICS

By

SAUMYA SUSAN SIMON

Bachelor of Science in Mechanical and Aerospace Engineering

Oklahoma State University

Stillwater, Oklahoma

2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2011

SIMULATION OF THE FLOW OF A SINGLE STRANDED DNA IN A CHANNEL USING DISSIPATIVE PARTICLE DYNAMICS

Thesis Approved:

Dr. Khaled A. Sallam

Thesis Adviser

Dr. Frank W. Chambers

Dr. Wei Yin

Dr. Sheryl A. Tucker

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

1. INTRODUCTION

1.1 General Statement of the Problem

1.1.1 Background

This study is concerned with simulating dispersed systems and solvent molecules simultaneously for mesoscopic flow systems with single stranded Deoxyribonucleic acid (DNA) molecules with potential application to DNA separation. Separation of DNA has significant importance in understanding the genome of an organism for genetic engineering and DNA profiling for forensics. The most common method for separation is gel electrophoresis but it limits the separation for DNA strands up to 40kbp (Pan, Ng, Li, & Moeendarbary, 2010). It may also take up to several days or weeks for longer strands. Many researches have proposed ideas that involve micro and nano systems. Hydrodynamic forces, electric fields and magnetic fields are most commonly used for separation of DNA from its medium in microchannels (Huber, Markel, Pennathur, & Patel, 2009), (Jellema, Mey, Koster, & Verpoorte, 2009), (Kang et al., 2009), (Perkins, Smith, Larson, & Chu, 1995) and (Smith, Babcock, & Chu, 1999).

Some of these systems separate DNA using entropic trappings where the flow separates DNA strands based on its length (Pan et al., 2010). Huber et al. (2009) and Jellema et al. (2009) used electrokinetic separation of DNA in a nanochannel. Electrokinetic separation includes capillary electrophoresis, gel electrophoresis and electrodes arrays and the flows are created by ionized medium or ionized particles inserted in the flow. Jellema et al. (2009) also used hydrodynamic forces created by the converging channels to provide pressure-driven flow along with electrokinetic flow. Kang et al. (2009) investigated a method of mixing magnetic particles that is controlled by magnetic field to separate DNA of a specific size.

Many methods have been created to simulate the flow of dispersed systems such as colloidal suspension in micro-systems and macro-systems. These molecular simulation methods simulate micro scale flows and translate them into their macroscopic counterparts. Dissipative Particle Dynamics (DPD) can provide an accurate simulation of a colloidal suspension at a mesoscopic scale with lesser computational cost and time steps (Symeonidis, Karniadakis, & Caswell, 2005) than other micro scale methods. Mesoscopic scale is an intermediate length scale which is gernerally considered to be between a few hundred nanometers and a micrometer. It consists of a large number of atoms but takes quantum effects into account. Hoogerbrugge and Koelman (1992) initiated DPD to simulate such an experiment with lesser computational cost and time steps. . Even though DPD does not replicate the correct molecular motion at the atomic level, it does provide accurate hydrodynamic properties for dispersed systems for long lengths and large time steps (Frenkel & Smit, 2002). Mesoscopic scales, long time steps and colloidal suspension should be taken into account when one tries to understand the

flow dynamics of a single strand DNA molecule through microchannels (Fan, Phan-Thien, Chen, Wu, & Ng, 2006).

1.1.2 Problem Statement

Simulation of DNA can provide insight into its physical properties without costly and demanding experimentation. This research is concerned with the flow of DNA molecules suspended in a solvent through a pressure-driven microchannel using Dissipative Particle Dynamics. The goal is to understand the changes in the mechanical properties of these flows, including DNA stretching and migration. In the present study, DNA would be replicated as worm-like chain polymers. This study is of interest due to its applicability to the development of lab-on-a-chip for disease testing kits. Lab-on-a chip is a biomicroelectromechanical device (BioMEMS) which can be used for drug delivery and DNA testing for diseases. The computational methods employed in this study can also be used to simulate other colloidal suspensions such as liposome suspension, polymer interaction etc.

1.2 Previous Studies

When it comes to particle simulation, all forces acting on a particle including particle-particle interaction and particle-boundary interaction must be taken into consideration. There are different computational methods that can be used for such simulations (Satoh, 2011). Some of the most commonly used methods are Molecular Dynamics (MD), Brownian Dynamics simulation (BDS), lattice Boltzmann (LB), lattice

gas automata (LGA), and Dissipative Particle Dynamics (DPD) (Fan et al., 2006). Many of these methods are restricted to certain test conditions. Certain methods can only be used to simulate simple fluids, where others can be used only for simulating macro-scale systems. A brief discussion of these methods is presented first to explain why Dissipative Particle Dynamics was chosen to simulate a system of DNA strands suspensions. The related studies of DNA flow dynamics are reviewed next.

1.2.1 Molecular Dynamics (MD)

Molecular Dynamics (MD) has existed since the beginning of digitalization. It was created to simulate large celestial bodies to the minutest particle with the same Newton's Law. The equations may have been modified to accommodate newer discoveries in terms of physical properties, but it is nevertheless, a tried-and-true process that will be utilized for a long time to come.

MD is governed by the basic Newton's second law of motion. Consider a particle *i* with m_i as the mass, r_i as the position and f_i as the sum of the interactive forces between the particle and its environment, then the motion of particle is controlled by

$$m_i \frac{d^2 r_i}{dt^2} = f_i \qquad 1$$

To numerically solve the equations for N particles, a scheme called the Verlet method can be used. Such schemes can be used to find both positions and velocities of all N particles at different time-steps. These methods will be discussed in detail later.

MD simulations have been used for an assortment of studies. These may range from fundamental physics to phase changes to different molecular structures. A list of a few related studies was listed in *The Art of Molecular Dynamics Simulation* by Rapaport (2004). He showed that MD can be utilized for various purposes especially at different scales, phases, complex or simple structures, long or short ranged and so forth (Rapaport, 2004). MD can also bind the atoms together in case of solid or liquids if the atoms travel over a certain distance. Lennard-Jones potential can be used to create this interaction within MD simulations.

MD, however, cannot be used to correctly simulate systems that concerns quantum fluctuations at the atomic level. The softer interactions between particles lead to smaller time-step and higher internal motion. Small systems also increase fluctuations and limit the accuracy and the shape of the simulation region and the atomic trajectories may be unstable (Rapaport, 2004). Due to the higher computational cost, simulation is limited to simple fluids in two-dimensional system. MD can be used to model a simple flow in microscopic level and then translate it into macroscopic levels (Hoogerbrugge & Koelman, 1992). Therefore, MD may have a lot of potential but it cannot be utilized fully unless there are more computational advancements. Also, classical theories such as quantum mechanics are still in theoretical state when it comes to simulation (Frenkel & Smit, 2002).

1.2.2 Monte Carlo (MC)

Monte Carlo (MC) is similar to Molecular Dynamics as it can be used for simulations at a microscopic level and then the results are used to provide information on

properties at the macroscopic scale. MC operates under a stochastic law and generates different microscopic states. Since it does not follow the equations of motion, it does not follow the changes with respect to time. This limits this method to applications with systems at thermodynamic equilibrium and thus, it is unsuitable for dynamics systems as time cannot be conceived.

Consider state 1 where two particles are overlapping; this will create a repulsion force between the particles and an interactive energy will rise. In state 2, two particles are at a close proximity where the repulsion has decreased and attraction forces have started on the particles. State 3 has two particles at a distance where their interaction is negligible and the energy is very low. In actual systems, microscopic states with high energy such as state 1 rarely exist; instead, state 2 with low energy and weaker interaction forces are more applicable. These states give rise to a minimum free energy of the system. This can be seen from a system with temperature T, volume V, and number of particles N where the Helmholtz free energy F becomes a minimum (Satoh, 2011).

$$F = E - TS \qquad 2$$

where E is the potential energy and S is the entropy of the system. For example, if oxygen and nitrogen were to fill a room, the entropy will keep the energy in check of the minimum free energy of the system. To numerically evaluate this theory, one can use probability density function for N number of particles to find new positions at a set interval provided that N, V, and T are given. The probability functions depend on the interaction energies of the different states. The step by step algorithm to process MC is given in *Introduction to Practice of Molecular Simulation* by Satoh (2011).

1.2.3 Brownian Dynamics

Brownian Dynamics is used when a system contains dispersed particles in a liquid base. These systems cannot be modeled using MD or MC as they will generate the motion for the solvent particles based on their characteristic time and the dispersed particles will not be accounted. Also, the solvent molecules must be simulated as a continuum as compared to being computed individually. The motion of the solvent molecules will be reflected as a random force in the dispersed particles' equation of motion. BD simulates the random walk of the dispersed particles induced by the solvent particles. The particles moving due to the random force is called "Brownian particles" (Satoh, 2011).

Consider a dispersed solution that is generously diluted that the particles can be regarded as moving independently. Their motion can be analyzed using the Langevin equation (Satoh, 2011) as follows:

$$m\frac{dv}{dt} = f - \xi v + f^B \qquad 3$$

where m is the mass of a spherical particle, v is the velocity vector and ξ is a coefficient given by $\xi = 3\pi\eta d$ (d is the particle diameter and η is the viscosity of the solvent), f is the external force and f^B is the random force vector of the solvent. The random force has a zero mean and variance of $2\xi kT\delta(t-t')$ where δ is the Dirac delta function. The random force is proportional to the system temperature and thus the particles act vigorously in high temperature. Satoh (2011) explains random displacements and the procedure for BD simulation in detail.

1.2.4 Lattice Boltzmann

Lattice Boltzmann can be used for dispersed particle systems. The simulation region is a lattice network that contains virtual fluid particles that interact with each other. Fluid particles are assumed to be clusters of solvent particles that are allowed to move to its neighboring sites only.

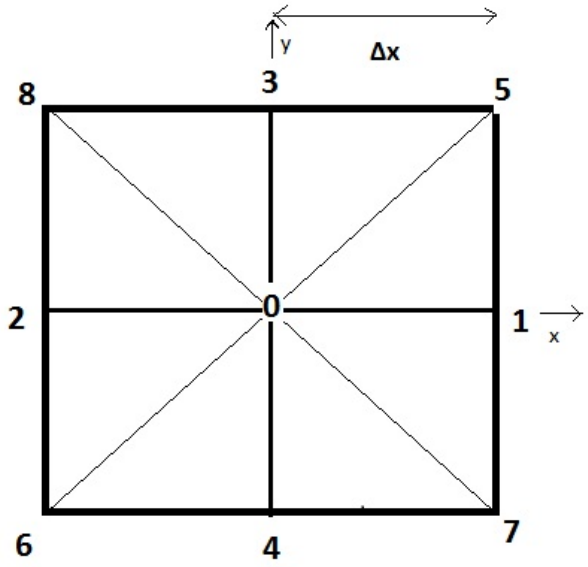

Figure 1 Lattice Boltzmann 2-D Lattice Model (D2Q9) (Satoh, 2011).

For example, the particle at point 0 can move to sites 1, 2, 3, 4, 5, 6, 7, and 8. If the particle is moving to 1, 2, 3, or 4, it will have a velocity c = Δx/Δt. If the particle is moving to sites 5, 6, 7, or 8, the velocity is $c\sqrt{2}$. Δx is the distance between two sites and Δt is the time interval. In a two-dimensional lattice with nine possible sites for the particle to move, including the original position, the model is called D2Q9 (Satoh, 2011).

The Boltzmann equation describes the temporal evolution of the velocity distribution function at all points. A digitized distribution, $f_a(r,t)$ is tracked in LBM where a is the direction, r is the position, and t is the time. The density at a location *r* at time *t* is

$$\rho(r,t) = \sum_{a=0}^{8} f_a(r,t) \quad 4$$

Similarly, the velocity is given by

$$\rho(r,t)u(r,t) = \sum_{a=0}^{8} f_a(r,t)c_a \quad 5$$

The velocity vector, c_a, is the velocity of particle moving to the neighboring sites in the direction a, where a=0, 1, 2…8. The collision term in Boltzmann equation is difficult to evaluate and models such as Bharatnagar-Gross-Krook (BGK) are used to simplify the expression enough to be solved (Satoh, 2011).

1.2.5 Dissipative Particle Dynamics

DPD has been used to simulate many mesoscopic systems, including liposome formation, colloidal suspension, red blood cell flow, concrete and other non-Newtonian substances, among others. The advantages of DPD are that it considers hydrodynamic behavior without additional formulation and it emulates the Brownian motion the particles follow. Moreover, unlike MD or MC, computational cost is cheaper.

Dissipative Particle Dynamics can simulate dispersed and solvent particles simultaneously, similar to BD, but with a different approach to the solvent particles. A set of solvent molecules is considered as one *virtual* fluid particle. Similar to the dispersed particles, the virtual particles have a corresponding characteristic time of the motion (Satoh, 2011). In this manner, one can simultaneously simulate the motion of both the

dispersed and fluid particles without having a secondary time-step for the fluid particles. These virtual particles will exhibit a similar random walk to that of the dispersed particles due to change in momentum and particle-particle interaction and potential changes. These virtual particles were then dubbed as dissipative particles.

DPD can simulate colloid suspensions such as polymer suspensions including all hydrodynamics forces. The equations of motion for DPD include these forces which can be used for spherical and non-spherical colloidal particles. DPD simulate mesoscopic systems as compared to macroscopic or microscopic systems. To simulate a fluid flow that follows Navier-Stokes equation, the total momentum should be conserved. Therefore, for a particle *i*, the total force acting on the particle consist of conservative or repulsive force, dissipative force and random force that provides the interparticle repulsive and attractive forces.

$$f_i^{int} = \sum_{j \neq i} F_{ij} = \sum_{j \neq i} F_{ij}^C + F_{ij}^D + F_{ij}^R \quad 6$$

This study deals with the simulation of dispersed particle such as DNA or polymers within a solvent or liquid base. Based on the present literature review, DPD is a suitable simulation method for the system in question. The forces, parameters and other equations required for DPD simulation is described in the next chapter.

1.2.6 Simulation of a Single Strand DNA

Perkins et al. (1995) measured the extension of tethered DNA strands in uniform pressure-driven flows. Smith et al. (1999) studied similar extension in a steady shear-flow. Extension of DNA has been studied using optical tweezers to restrain the DNA

strand at one end to understand its extension effects by (Perkins et al., 1995) and (Larson, Perkins, Smith, & Chu, 1997), among others. The studies of single strand DNA performed by these authors explained its dynamics and rheological properties and how it is closely related to the properties of polymer particles (Larson et al., 1997). This similarity allows the simulation of DNA strands as beaded polymers through different models such as worm-like chains, FENE and Hookean-Fraenkel. The bead forces include Lennard-Jones repulsion potential, FENE springs and worm-like chains (WLC) forces (Symeonidis, Karniadakis, & Caswell, 2006).

Fan et al. (2006) studied the flow of a single stranded DNA through a pressure-driven microchannel using Dissipative Particle Dynamics computational method. Using Hoogerbrugge and Koelman's (1992) proposed idea, Fan et al. (2006) studied DNA flow dynamics through a microchannel, made the appropriate modifications to improve the characteristics of the DPD method and the worm-like chain modeling of DPD particles. Fan et al. (2006) illustrated how low Schmidt number and inadequate viscosity can be corrected by increasing the cutoff radius or by reducing the exponent parameter *s*. The modifications to the weighting number were shown to provide the best result along with the least computational cost. Fan et al. (2006) suggested that changing the cutoff radius enhances Schmidt's number adequately, but it tends to increase computational cost by 2.6 times. Worm-like chain was modeled with a large number of beads and a weak repulsive force of DPD illustrated in Figure 2.

(Fan et al., 2006) study simulated the physics of DNA folding, unfolding, entanglements and extension of the strands to understand molecular structural changes as external forces are exerted on strands with varying number of beads. Their computational

results of the extension of the beads agreed well with the experimental data provided by Perkins et al. (1995). They investigated the DNA extension in a uniform flow which was generated by not employing any numerical means to slow down the flow of the near-wall particles. However, the DNA extension in a microchannel with pressure-driven Poiseuille flow is more practical and needs further attention. This is applicable in many biological devices that can be used for DNA delivery. Devices such as microneedles and array of hollow microcapillaries were used by Chun et al. (1999) for controlled injection of DNA into cells. Understanding the effect of DNA migration and extension in microchannels will increase the efficiency of such devices.

To acquire accurate results, the boundary conditions need to be implemented to sustain a long channel and solid wall replications. Studies have been performed to understand the different types of boundary conditions and the methods to apply them (Revenga, Zúñiga, & Español, 1999); (Revenga, Zúñiga, Español, & Pagonabarraga, 1998); and (Pivkin & Karniadakis, 2005). Many of these authors suggests the reflection of the particles from the walls by reversing the normal velocity vectors, assigning random velocity and injecting the particles towards the flow, or reversing both tangential and normal velocity vector of the particles at the wall. The wall must be in tight lattice with layering and higher density. However, using just these measures may not prevent particles from penetrating the wall. A no-slip boundary condition has been proposed where a layer of DPD fluid layer is inserted next to the frozen particle wall. This is a practical method but further modifications to the boundary condition are still necessary as the soft repulsion of DPD particles may not prevent the penetration of the walls.

1.3 Specific Objectives

The present study investigates numerically the dynamics of a single strand DNA in a pressure-driven channel flow. A program was developed using Matlab (a Mathworks product) to conduct the present simulation. The computational methods are first validated by simulating Poiseuille flow through a microchannel. The DNA flow is then simulated for various test conditions. The list of specific objectives is as follows:

1. Develop a computer program to conduct a DPD simulation of a Poiseuille flow. Validate the present computational methods using previous results from Fan et al. (2006, 2003) and Symeonidis et al. (2005).
2. Investigate the effect of the number density and the weighing function on Poiseuille flow simulation.
3. Modify the boundary conditions for the DPD method to enforce a no-slip boundary condition while maintaining a simple solid wall construction.
4. Simulate DNA particle flow through a pressure-driven channel and replicate its stretching and folding properties under different conditions such as the strand placement and the number density.

1.4 Organization of the Thesis

The thesis is organized into four chapters and appendices. The first chapter highlights the differences among the different simulation methods available and the reason that Dissipative Particle Dynamics have been chosen in the present work. The system considered is a colloidal suspension with polymers as the dispersed particles and

it is suspended in fluid particles or solvent molecules in reality. The first chapter also includes the statement of the problem and the specific objectives. The second chapter illustrates the formulation of DPD and describes the effect of each parameter on the output of the modeling. The results and discussions of the simulation are in the third chapter. This chapter compares the modified DPD flow to the theoretical values and provides the trajectories and extension properties of DNA or polymer suspension in fluid particles at different conditions. The last chapter of the thesis summarizes the results and conclusions and also provides recommendation for the betterment of the present computational methods. The appendices include the code and additional information used to help generate the algorithm.

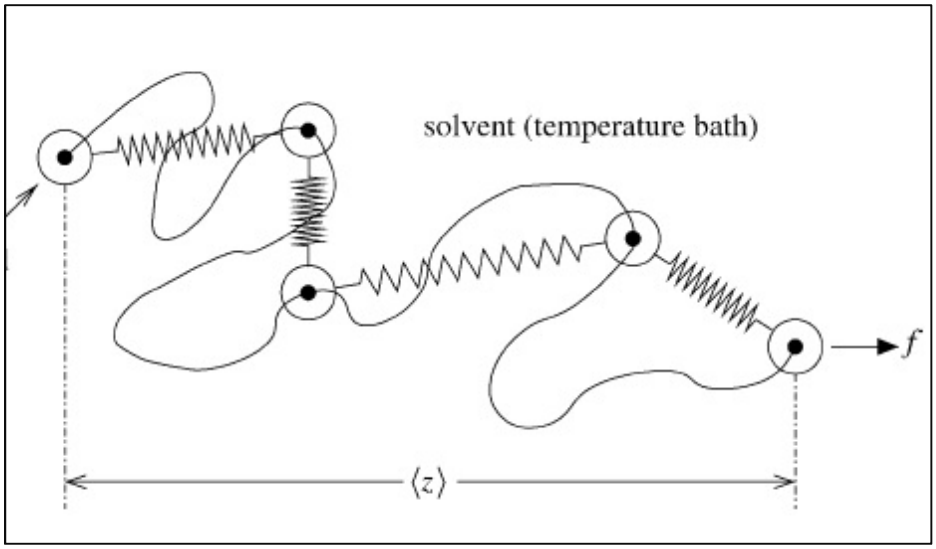

Figure 2 Worm-like-chain DNA Strands (Underhill & Doyle, 2004)

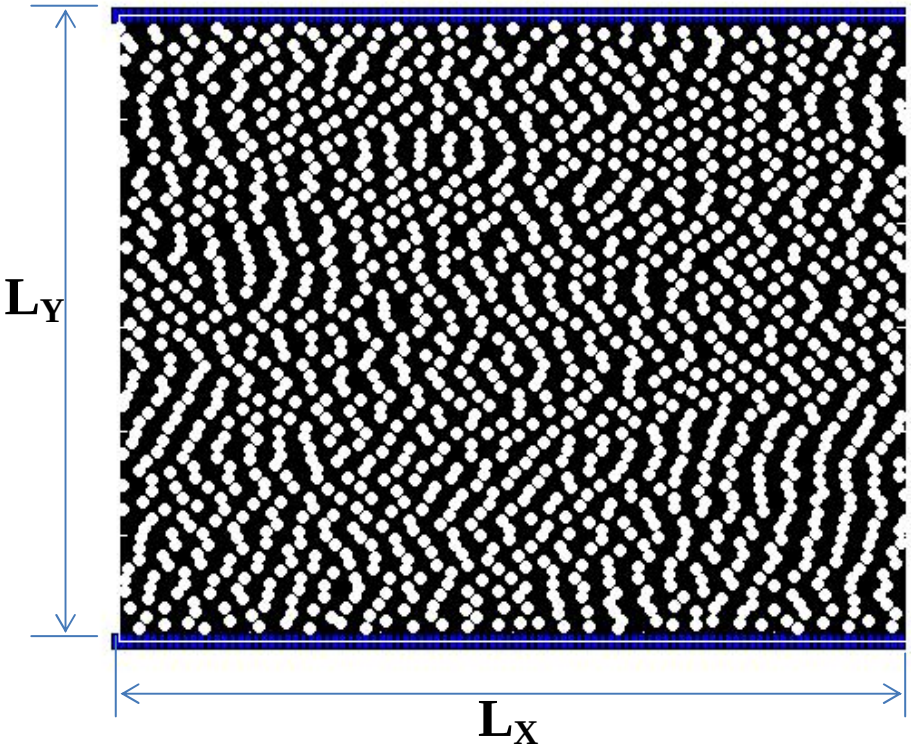

Figure 3 Simulation grid

CHAPTER II

2. COMPUTATIONAL METHODS

2.1 Dissipative Particle Simulation Theory

Dissipative Particle Dynamics follows the basic principle of Navier-Stokes equations with certain alterations. The simulated fluid systems are isotropic and Galilean invariant. The computational cost and time is much smaller than other simulation practices such as MD or Monte Carlo.

The original formulation was created by Hoogerbrugge and Koelman (1992) when they first introduced DPD in their study, *Simulating Microscopic Hydrodynamic phenomena with Dissipative Particle Dynamics*. The following stochastic differential equation was created to maintain the Brownian motion of the colloidal and polymer particles (Hoogerbrugge & Koelman, 1992) :

$$r_i' = r_i + \frac{\delta t}{m_i} p_i' \quad (7)$$

$$p_i' = p_i + \sum_j \Omega_{ij} e_{ij} \quad (8)$$

$$\Omega_{ij} = W\left(|r_i - r_j|\right)\left\{\Pi_{ij} - \omega\left(p_i - p_j\right) \cdot e_{ij}\right\} \quad (9)$$

where Ω_{ij} is the weighted function that would balance the system from over-fluctuation and over-relaxation. The first part of the weighted equation corrects the pressure effects whereas the damping part introduces the viscosity effects (Hoogerbrugge & Koelman, 1992). This is taken as a coarse-grained system and it follows the Navier-Stokes equations of continuum flow.

Consider a system with N colloidal particles having equal mass, where m_i =m=1, with positions r_i and velocities v_i. Fan et al. (2003) found that the simple DPD fluid behaves as a Newtonian one. The changes of positions and velocities as time evolves are determined by basic Newton's laws:

$$\frac{dr_i}{dt} = v_i \tag{10}$$

$$\frac{dv_i}{dt} = f_i = f_i^{int} + f_i^{ext} \tag{11}$$

The total force, f_i, includes internal, f^{int}, and external forces, f^{ext}. The internal forces consist of all inter-particle forces between fluid-fluid, fluid-wall, fluid-polymer, polymer-polymer and polymer-wall. The external force could be a gravitational, an electrical, or a magnetic force. In this study, we have chosen a gravitational force as the external force similar to (Liu, Meakin, & Huang, 2007). There are three internal forces exerted on particle *i* by surrounding particles *j*. These forces, discussed below, are the conservative repulsive force, F^C, the dissipative force, F^D, and the random force, F^R.

$$f_i^{int} = \sum_{j \neq i} F_{ij} = \sum_{j \neq i} F_{ij}^C + F_{ij}^D + F_{ij}^R \tag{12}$$

The conservative repulsive force, F^C, given by:

$$F_{ij}^{C} = \begin{cases} a_{ij}\left(1 - r_{ij}/r_c\right)\hat{r}_{ij}, & r_{ij} < 1, \\ 0, & r_{ij} \geq 1, \end{cases} \quad 13$$

provides adequate repulsion to both fluid and polymer particles. DPD is a soft core system. A soft core system has weak repulsive forces between the particles and as a result the particles may overlap with each other. The parameter a_{ij} is the maximum repulsion factor between particles i and j. This parameter enables us to set repulsion strength between fluid-fluid, fluid-polymer and polymer-polymer interactions. $\mathbf{r_{ij}} = \mathbf{r_i} - \mathbf{r_j}$, $r_{ij} = |\mathbf{r_{ij}}|$, and $\hat{r}_{ij} = \mathbf{r_{ij}}/r_{ij}$ is a unit vector that determines the direction from *j* to *i*. The conservative force depends on the ratio between r_{ij} and a cutoff radius, r_c, which is the length unit in this study. The length of the channel and the distance travelled by the fluid particle are given in terms of this cutoff radius.

The dissipative and random forces are given by:

$$F_{ij}^{D} = -\gamma w^{D}(r_{ij})(\hat{r}_{ij} \cdot v_{ij})\hat{r}_{ij} \quad 14$$

$$F_{ij}^{R} = \sigma w^{R}(r_{ij})\theta_{ij}\hat{r}_{ij} \quad 15$$

where γ and σ are characteristic strength of each force respectively. Their relation is given by:

$$\gamma = \frac{\sigma^2}{2k_B T} \quad 16$$

where k_B is the Boltzmann's constant and T is the temperature of the system. For simplicity, k_BT is taken as a unit value. These parameters provide a stable standard DPD

simulation and relax any unusual fluctuations. The factors w^D and w^R are r-dependent weighting functions which are computed according to the neighboring j particles, $\mathbf{v_{ij}} = \mathbf{v_i} - \mathbf{v_j}$ and θ_{ij} is white noise function given by (Fan et al., 2006):

$$\langle \theta_{ij}(t) \rangle = 0; \quad \langle \theta_{ij}(t)\theta_{kl}(t') \rangle = (\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\delta(t - t') \qquad 17$$

For simulation purposes θ_{ij} is also a random number from a Gaussian distribution which has zero mean and unit variance (Willemsen, Hoefsloot, & Iedema, 2000). In this study, we have used the Maxwell-Boltzmann distribution, given below, to generate random numbers as θ_{ij}.

$$f(\vec{v}) = \left(\frac{\beta}{\pi}\right)^{3/2} e^{-\beta v^2} \qquad 18$$

All three forces are dependent on distance and F^D and F^R are functions of velocities as well. The forces conserve the linear and angular momentum and they act along the line between two particles. The weighting functions are given as:

$$w^D(r_{ij}) = \left[w^R(r_{ij})\right]^2 = \begin{cases} (1 - r_{ij}/r_c)^s, & r_{ij} < r_c \\ 0, & r_{ij} \geq r_c \end{cases} \qquad 19$$

The weighting functions, w^D and w^R, are zero when $r_{ij} \geq r_c$, thus calculating only forces for particles j within the radius r_c around the particle i with an exponent s. When $r_c = 1$ and $s = 2$, the basic DPD quadratic weighting function, $(1-r)^2$, is acquired. However, the cutoff radius and exponent can be changed, $r_c \geq 1$ and $s \leq 2$, to modify particle interaction.

s can be decreased or r_c can be increased to increase particle interactions. Modifications to exponent *s* are considered more applicable as increasing the cutoff radius increases computational cost. As the cutoff radius is increased, the number of neighboring particles rises, and the simulation time increments. In this study we explore the effects of varying the exponent *s*.

The viscocity of DPD fluids includes contribution from the diffusion motion of the particles and the dissipative forces (Groot & Warren, 1997).

$$\eta = \frac{\rho D}{2} + \frac{2\pi\gamma\rho^2 r_c^5}{15}\left(\frac{1}{s+1} - \frac{4}{s+2} + \frac{6}{s+3} - \frac{4}{s+4} + \frac{1}{s+5}\right) \qquad 20$$

The Schmidt number is the ratio between dynamic viscosity and diffusion rate of the fluid, $Sc = \eta/\rho D$. The viscosity and diffusion correlations for s=2 and s=0.5 is given in Table 5. The parameters of the DPD fluid can be adjusted to match the pgysical properties of the solvent fluid. Table 5 in Appendix illustrates the relationship between the DPD parameters and the viscosity, diffusitivity and the Schmidt number of the solvent. Similar to Fan et al. (2003) the unit of length $[r_c]$ = 1.608 μm, the unit of velocity is [V]= 0.345 cm/s and the unit of time, t = $[r_c]/[V]$ is 4.661 x 10^{-4}s. We used a time step of 0.02 or in dimenaional unit of 0.93 x 10^{-5}s.

2.2 DPD Integration Methods

As time evolves, the new particle trajectories and velocities have to be determined by Newton's laws. There are many methods that have proved to be efficient and yield accurate values of the new positions and velocities. Some of the methods are given next.

2.2.1 Euler's Method

The simplest method is the Euler's method where the new particle positions and velocities at time, t+Δt, are derived from the previous position and velocity at time t.

$$r_i(t + \Delta t) = r_i(t) + \Delta t v_i(t) \quad 21$$

$$v_i(t + \Delta t) = v_i(t) + \Delta t F_i(t) \quad 22$$

The force is calculated, following the above equations, using the new position and velocity.

$$F_i(t + \Delta t) = F_i(r_i(t + \Delta t), v_i(t + \Delta t)) \quad 23$$

However, this method causes an energy drift and yields particle trajectory that is not time reversible. Energy drift is the gradual increase of the total energy of the system due to numerical inaccuracies and energy fluctuations. However, total energy of the system is theorectically constant according to the laws of physics. DPD can resolve the energy drift problem (Pivkin, Caswell, & Karniadakis, 2011).

2.2.2 Verlet-type Algorithm

Another method is the Verlet-type. It uses only the new positions to calculate the inter-particle forces at different time steps. This method uses positions at t and (t-Δt) (Pivkin et al., 2011) as follows:

$$r_i(t+\Delta t) = 2r_i(t) - r_i(t-\Delta t) + \frac{1}{M}(\Delta t)^2 F_i(t) \quad 24$$

$$F_i(t+\Delta t) = F_i(r_i(t+\Delta t)) \quad 25$$

where M = m = 1, is a unit mass of the particles. The velocity is not calculated for this scheme and the force is position dependent only. Due to this reason, this method is not a good fit for this study as DPD forces need instantaneous velocity values for their calculations.

2.2.3 Velocity-Verlet Algorithm

The Velocity-Verlet method is an extension of the Verlet-type algorithm to predict the velocity of the particle at the new position using velocity at (t) and the force calculated using the new position. The force only uses the instantaneous positions for its calculations.

$$r_i(t+\Delta t) = r_i(t) + \Delta t v_i(t) + \frac{1}{2}(\Delta t)^2 \frac{1}{M} F_i(t) \quad 26$$

$$F_i(t+\Delta t) = F_i(r_i(t+\Delta t)) \quad 27$$

$$v_i(t+\Delta t) = v_i(t) + \frac{1}{2}\Delta t \frac{1}{M}[F_i(t) + F_i(t+\Delta t)] \quad 28$$

2.2.4 Modified Velocity-Verlet for DPD

The changes in the Velocity-Verlet algorithm made by Groot and Warren (1997) take the new position and predictive velocity into consideration while determining the

new force acting on the particle. With this force, the actual velocity is computed to determine the next trajectory of the particle. The changes are as follows.

$$r_i(t+\Delta t) = r_i(t) + \Delta t v_i(t) + \frac{1}{2}(\Delta t)^2 \frac{1}{M} F_i(t) \quad 29$$

$$\tilde{v}_i(t+\Delta t) = v_i(t) + \lambda \frac{1}{M} F_i(t) \quad 30$$

$$F_i(t+\Delta t) = F_i(r_i(t+\Delta t), \tilde{v}_i(t+\Delta t)) \quad 31$$

$$v_i(t+\Delta t) = v_i(t) + \frac{1}{2}\Delta t \frac{1}{M}[F_i(t) + F_i(t+\Delta t)] \quad 32$$

where λ is the variable that will utilize the effects of the stochastic processes (Pivkin et al., 2010). According to Groot and Warren (1997) when λ=1/2, the system should relax to the Velocity-Verlet algorithm. Since the DPD forces uses velocity in its calculation, the predictive velocity,$\tilde{v}_i$, can be used to do so.

Due to the change in modified velocity-Verlet algorithm, the random velocity is also modified to be effected by the varying time step. According to Groot and Warren (1997), the change in random force should be independent of the time-step as diffusion relies upon this force greatly. Diffusion needs to be independent of the time-step and thus random force is accurately computed when divided by $\sqrt{\Delta t}$ (Groot & Warren, 1997). However, Fan et al. (2006) and other authors have ignored this modification. In this study, we have conducted simulation with the modification to the random force.

$$F_{ij}^R = \sigma w^R(r_{ij})\theta_{ij}\Delta t^{-\frac{1}{2}}\hat{r}_{ij} \quad 33$$

The random force is divided by the time step so that the velocity calculation will not have to be modified. The velocity is initially as follows:

$$v_i(t+\Delta t) = v_i(t) + \frac{1}{2}\Delta t \frac{1}{M}\left[F_{C,D}(t) + F_{C,D}(t+\Delta t)\right] + \frac{1}{2}\sqrt{\Delta t}\frac{1}{M}\left[F_R(t) + F_R(t+\Delta t)\right] \quad 34$$

If the random force is modified while the forces are computed, the velocity-Verlet need not be changed later. In this manner, the diffusion is corrected in the simulation.

2.3 Initial Conditions

2.3.1 Initial Positions

The particles are initially allocated equidistantly. Randomly assigning positions for particles tend to increase overlap of particles. The allocations are either simple cubic lattice, face-centered cubic lattice or body-centered cubic lattice. Similar formation can be utilized for two-dimensional as well as three-dimensional configuration (Satoh, 2011).

The configuration in Figure 4-A is the basic cubic lattice which can be used for gaseous particles. The particles are separated by a distance of 'a' which is usually equal to the particle diameter. This is inappropriate for liquid or solid particles as there is only one particle in a unit cell. For a system with N particles, where $N = Q^2$, a square unit cell with sides (Q-1) × (Q-1) can be generated with side length, $L = Qa$. Thus, the number of particles, N, should be a square of a natural number such as 1, 4, 9, etc. the number density, n, is given by N/L^2. For a simulation, the N and n is initially set from which Q, L and packing distance, a, can be determined.

Figure 4-B has a higher packing density and therefore can be used for gaseous and liquid particles and limited solid systems. This lattice contains 2 particles per unit cell

and where $L = Qa$ and the square cell is built with (Q-1) × (Q-1) sides. N can only be of size 2, 8, 18, etc., and the density is $n = N/L^2$.

Figure 4-C has the most compact lattice for a 2-D system and can be used for solid systems as well. Each unit cell has 4 particles and thus $N=4Q^2$ with N = 4, 16, 36, etc. Each unit lattice can be replicated (Q-1) times each side to create the whole lattice with side lengths $L_x = 3^{1/2}Qa$ and $L_y = 2Qa$ and density $n = N/ L_x L_y$.

For the simulation conducted in this study, we use the lattice constructed with 2 particles in each unit since the simulation will contain mainly fluid particles and nano-sized polymers.

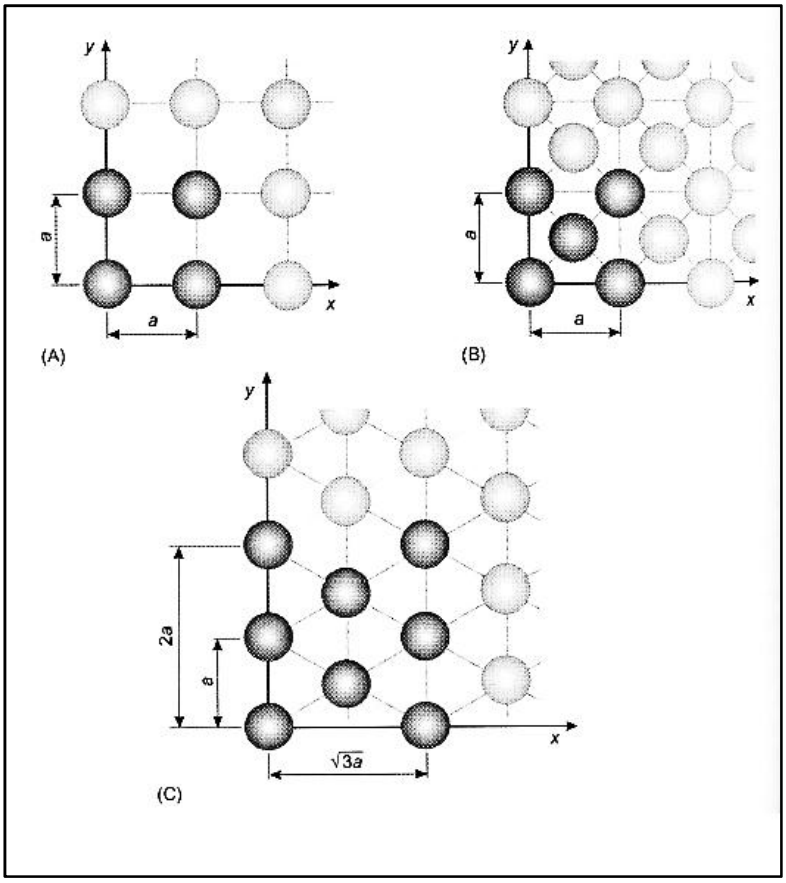

Figure 4 Configuration of Initial Condition of 2-D simulations (Satoh, 2011).

2.3.2 Initial Velocities

The initial velocities of the DPD particles, fluid or polymer, are assigned randomly with the Maxwell-Boltzmann's distribution. This distribution is selected so as to provide velocities with a zero mean. Considering a system with thermodynamic equilibrium with constant temperature T, the following Maxwellian distribution is employed to determine the particle velocities (Satoh, 2011).

$$f(v_i) = \left(\frac{m}{2\pi k_B T}\right)^{3/2} exp\left\{-\frac{m}{2k_B T}\left(v_{ix}^2 + v_{iy}^2 + v_{iz}^2\right)\right\} \quad 35$$

where k_B is the Boltzmann's constant, T is the temperature, m is the equal mass of the particles, and $v_i = (v_{ix}, v_{iy}, v_{iz})$ are the velocity vectors of particle *i*.

In order to create random placements of particles in DPD simulations, it is required to create random velocities according to a particular probability distribution. We need to use a uniform random number generator from zero to unity. The following equation with the random numbers is called the Box-Muller method (Satoh, 2011).

$$\left.\begin{array}{l} v_{ix}=\{-2(kT/m)\, ln\, R_1\}^2\, cos(2\pi R_2) \\ v_{iy}=\{-2(kT/m)\, ln\, R_3\}^2\, cos(2\pi R_4) \end{array}\right\} \quad 36$$

where R_1, R_2, R_3 and R_4 are random numbers from a uniform sequence.

2.4 Boundary conditions

Based on the type of system, different methods can be constructed so that the particle stays within the simulated region and observes its physical properties. DPD is

used for mesoscopic systems and a few different methods can be used to set the boundary conditions.

1. Periodic Boundary Conditions

Figure 5 illustrates the periodic boundary condition for a two-dimensional system (Satoh, 2011) where this boundary condition is assigned in both the x- and y- axis. The original simulation region is marked by the length of the system by L_x and L_y. The schematic shows how the particles moving across the boundary surfaces appear in the opposite side of the region. Periodic boundary not only displaces the position in a continuous flow but also transfers the energy and velocity to the next region.

For a simulation region with the (0, 0) coordinate in the middle of the system, the periodic boundary condition can be administered as follows:

```
if r(x,i)>= LX/2
    r(x,i)= r(x,i)-LX;
elseif r(x,i)<= (-LX/2)
    r(x,i)= r(x,i)+LX;
end
```

where *i* denotes the particle under consideration and `r(x,i)` is the position of the particle in the x-direction. This can be performed in x-, y- and z- axis.

The periodic boundary condition should also be applied when the interaction between particles *i* and *j* is calculated for the DPD forces. It must accommodate other treatments such as cutoff radius to provide accurate computation of forces and other values across the boundaries.

```
if diffr(x,i) > LX/2
```

```
        diffr(x,i)=diffr(x,i)-LX;
    elseif diffr(x,i) < (-LX/2)
        diffr(x,i)=LX - abs(diffr(x,i));
    end
```

where diffr(x,i) is the center to center distance between particles *i* and *j* whom the forces will act upon.

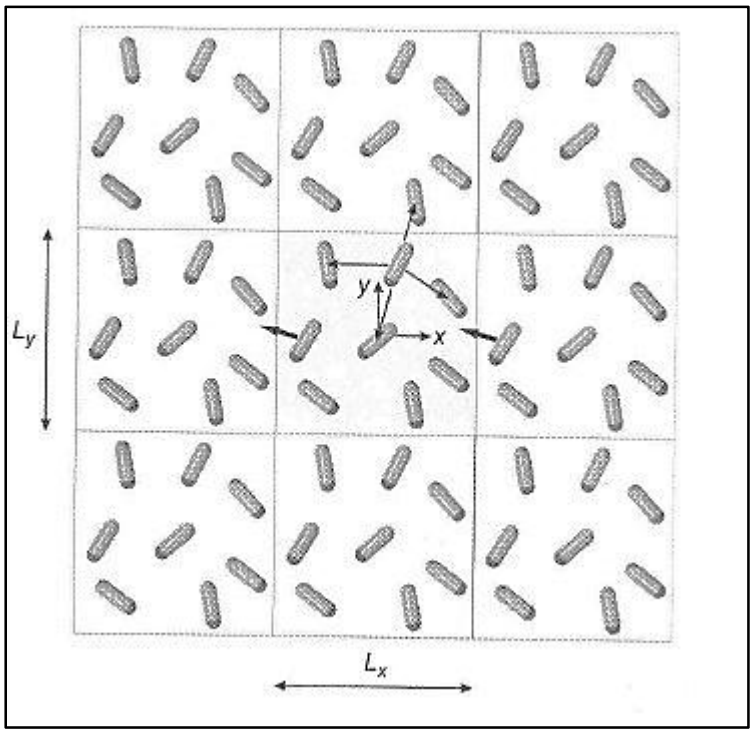

Figure 5 Periodic Boundary Condition in x- and y- directions (Satoh, 2011).

2. Lees-Edwards Boundary Conditions

Lees-Edwards method is used when non-equilibrium systems needs to be simulated. One of the simplest examples is the simple shear flow found in Couette systems. Couette flows have moving walls with constant velocities, U and –U, which provides uniform shear flow across the channel.

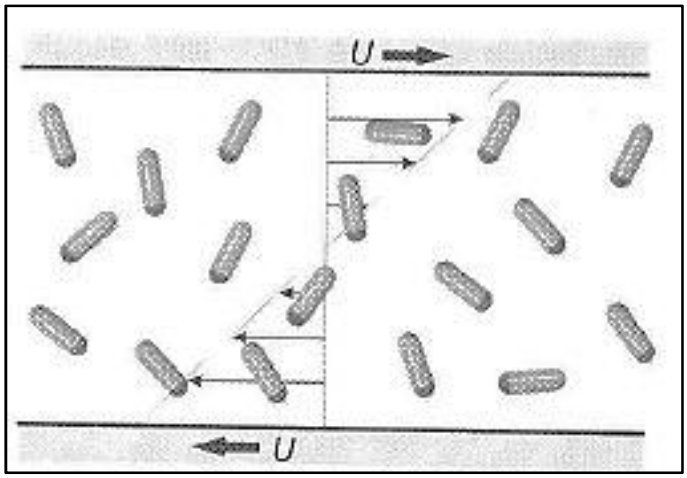

Figure 6 Simple Shear Flow in Couette Flow (Satoh, 2011).

To accommodate this system, the replicated boundary regions are shifted slightly in opposite directions by a distance ΔX with respect to the movement of each wall. This is depicted in figure 4. When the new positions and velocities are calculated, the change shifted distance, ΔX, and wall velocity, U, has to be considered. The particles moving out of the region in the x-direction will be shifted from x to (x - ΔX) and v_x to (v_x - U). The y direction will follow the same procedure as the periodic boundary condition. For example, the loop that can be used to set the boundary condition is as follows:

```
if r(x,i)>= LX/2
    r(x,i)= r(x,i)-LX-delX;
```

```
        v(x,i)= v(x,i)-U;
    elseif r(x,i)<= (-LX/2)
        r(x,i)= r(x,i)+LX+delX;
        v(x,i)= v(x,i)+U;
    end
```

where delX is the shift of the boundary region, ΔX and `v(x,i)` is the velocity of the particle *i* in the x-direction. For calculating the inter-particle forces between particles *i* and *j*, a similar loop with the same treatment is used as follows:

```
    if diffr(x,i) > LX/2
        diffr(x,i)=diffr(x,i)-LX -delX;
    elseif diffr(x,i) < (-LX/2)
        diffr(x,i)=LX - abs(diffr(x,i))+ delX;
    end
```

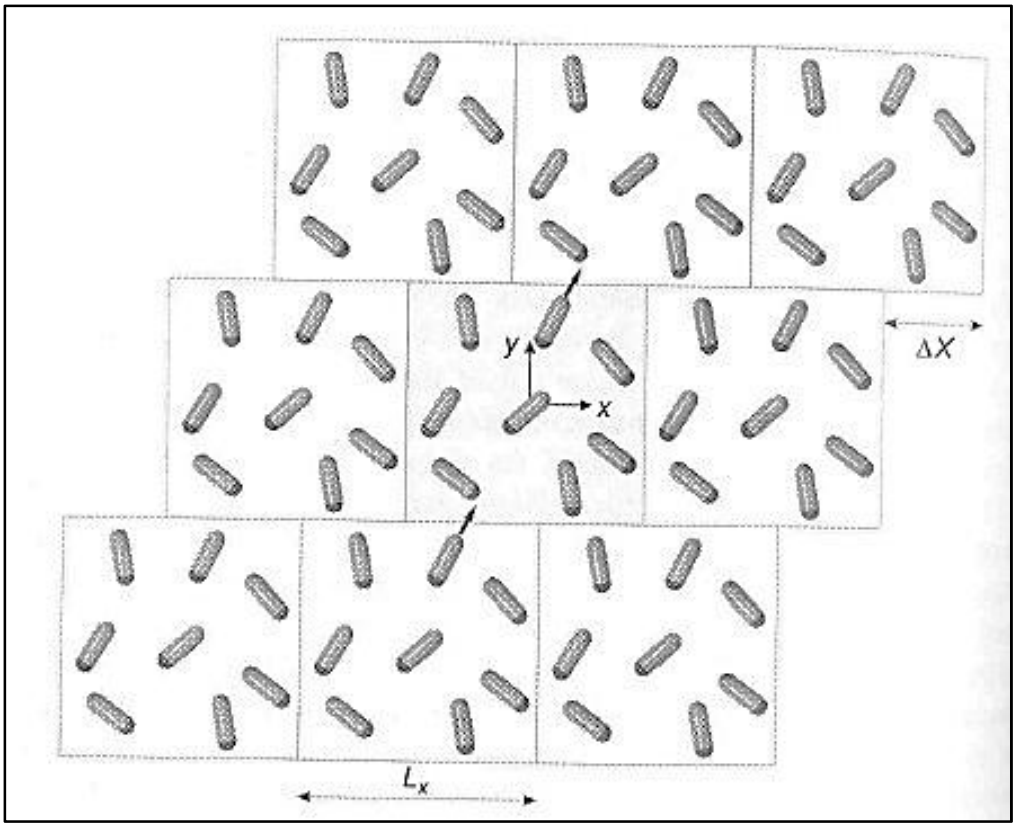

Figure 7 Lees- Edwards Boundary Condition for Shear Flow (Satoh, 2011).

3. Wall Boundaries

Solid boundaries tend to affect the density fluctuations, the loss of temperature, and unlike MD simulations soft repulsion in DPD particles does not prevent particles from crossing solid boundaries. Frozen particles acting as solid wall as well as multi-layered solid and fluid DPD wall particles can create boundary conditions that can be used for a Navier-Stokes continuum flow.

The first method is to group and create frozen DPD particles modeling a solid wall boundary for region under simulation. The particles could simply be a subset of the original lattice cell created for fluid particles as shown in Figure 8. The velocities for

these particles are zero so that they are constrained at a set position. However, due to soft repulsion, fluid particles can still penetrate these walls. Therefore a higher density packing is recommended increasing the repulsive force of the wall. Since the density of the wall is greater than the fluid particles, a no-slip condition is observed. Previous authors recommend a density ratio of one to nine between fluid and wall. However, as density increases repulsion grows which in turn causes density fluctuations at the wall (Willemsen et al., 2000).

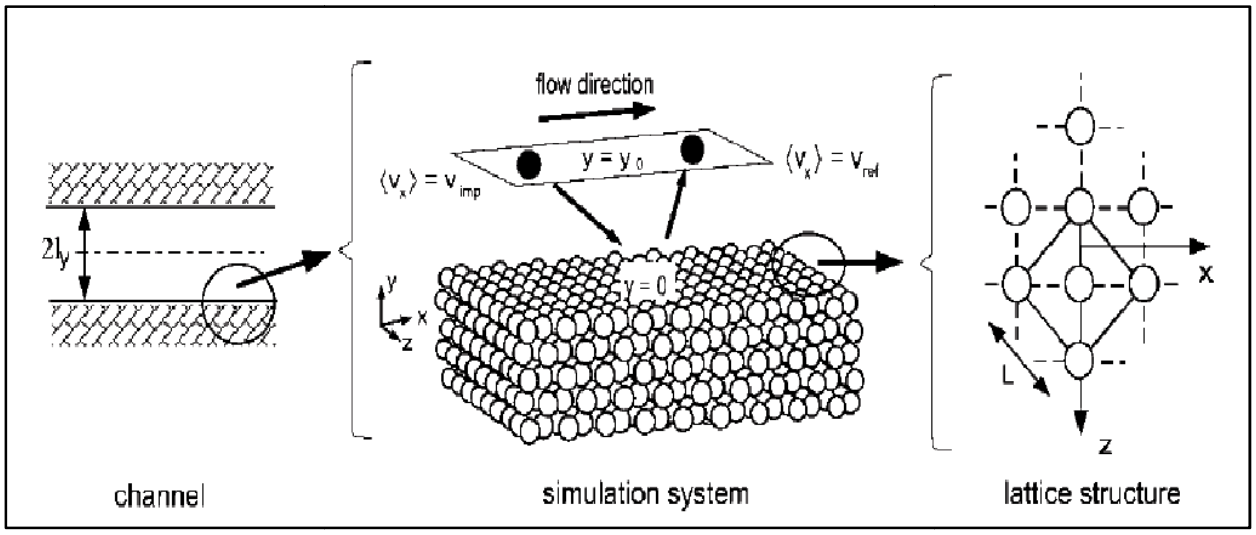

Figure 8 Lattice Wall (Arya, Chang, & Maginn, 2003).

The second method is to use frozen particles as a solid wall and to create a layer of fluid particles near the wall (i.e., wall fluid particles). The channel fluid particles crossing the wall fluid particles are reflected so that they would not penetrate the solid wall. Revenga et al. (1999) describe three different possibilities for reflecting these wall fluid particles as follows:

(1) A reflection of particles where its tangential velocity vector is conserved and the normal velocity vector is reversed, i.e. a specular reflection.

(2) Maxwellian reflections where the particles are assigned random velocity according to the Maxwellian distribution and send back into the flow i.e. diffuse reflection.

(3) Reflection of particles when both the tangential and normal velocity vector are reversed. This is called bounce-back reflection.

In this study, we have selected the Maxwellian reflection. No-slip Boundary conditions are created using random velocity vectors that shoot the particles back into the flow as they approach the wall. A no-slip boundary layer with thickness (from the wall) was set equal to 0.5% of the channel height. If the height of the wall is large such that 0.5% of the height is greater than the cutoff radius, the boundary layer thickness will be the cutoff radius. The purpose of this thickness is to prevent the cool down of fluid particles reaching the wall. The newly assigned velocity of the particle entering this layer will be as follows (Fan et al., 2006):

$$v_i = v_R + n\left(\sqrt{(n \cdot v_R)^2} - n \cdot v_R\right) \quad 37$$

where v_R is the random velocity with a zero mean and uniform distribution and n is the unit vector normal to the wall and pointing into the fluid channel.

2.5 Computational Efficiency

The force calculations take the most computational time within the DPD calculation. This cost limits the simulation to be of a small region or a two-dimensional

system. The force script runs N^2 times for each time frame. To reduce the CPU time and cost, the N^2 computation needs to be reduced. The methods provided below will reduce the N^2 loop considerably, thus reducing cost. This is mainly performed by limiting the interaction between every single particle within the systems to just the neighboring particles. Among other things, certain factors such as the increase in cutoff radius will substantially increase the cost as well.

2.5.1 Verlet or Neighbor List

In this method, a secondary cutoff radius, r_v, is used where $r_v > r_c$. A list is created for each particle which includes all the nearby particles positioned within the radius r_v. The CPU time will be reduced when the force acting on each particle is calculated only using those particles within the particle's list. When the particle is moved further than $r_v - r_c$, the Verlet or neighbor list is re-calculated. This method can be used for Molecular Dynamics and Dissipative Particle Dynamics methods.

2.5.2 Cell List

The simulation region is divided into $Q_X \times Q_Y$ cells with each individual cell having a size $(L_X/ Q_X) \times (L_Y/ Q_Y)$. For example, in Figure 9 if $Q_X = Q_Y = 6$ and $L_X = L_Y = 12$, each individual cell would be of 2 x 2 units. At the beginning of the process, the particles are grouped into each cell where each cell size is less than the cutoff radius square. Each cell needs to be named and stored along with its particles name and position. During the inter-particle DPD force calculations, a particle *i* will be calculated

with other j particles within the same cell and the neighboring cells. For example, a particle within cell 22 will be computed with other particles in 22 as well as 15, 16, 17, 21, 23, 27, 28, and 29. This method is very useful for large values of Q_X and Q_Y.

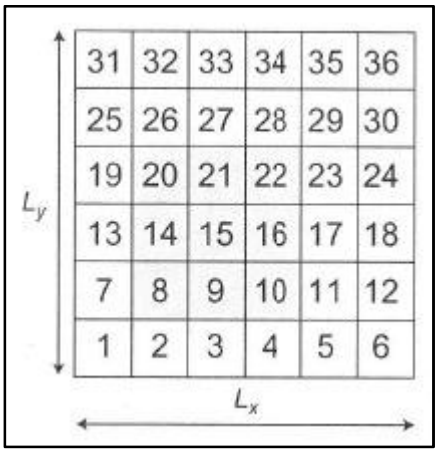

Figure 9 Cell List Method to Group Neighboring Particles (Satoh, 2011).

2.5.3 Cutoff Radius

The computational time is reduced when the number of particles used to calculate the forces are reduced. For a spherical particle, the Lennard-Jones potential, U_{LJ}, gives the interaction between two particles depending on the distance between them, as follows:

$$U_{LJ} = 4\epsilon\left\{\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right\} \quad 38$$

where σ is the characteristic length scale proportional to the particle diameter, r is the distance between the particles and $\epsilon = k_B T$. The ratio between σ and r determines the repulsive or the attractive behavior between the two particles. The interaction between particles can be negligible when r is greater than 3σ. The distance after which the

interactions between the two particles are too weak to be considered is called the cutoff radius.

In the simulation, an area with the cutoff radius, r_C, is created around each particle and forces are determined only between that particle and those particles located within that area. In this study we have chosen the cutoff radius as a primary method to reduce the computational cost. The value of r_C will be unity to acquire standard DPD properties. All the lengths within the simulation region will be given in terms of r_C.

2.6 DNA Modeling

Physical properties of DNA have been studied by Smith et al. (1992) and Perkins et al. (1995), among others. In these experiments the DNA had to be tethered to the wall or optical tweezers had to be used to gain knowledge of properties such as effect of shear on the fractional extension of DNA strands, and the effect of drag. These strands were also modeled using molecular simulation. DNA can be modeled within DPD as polymers. Figure 10 illustrates a model of DNA particles (tethered bead-chain particles) in DPD solvent particles (dots).

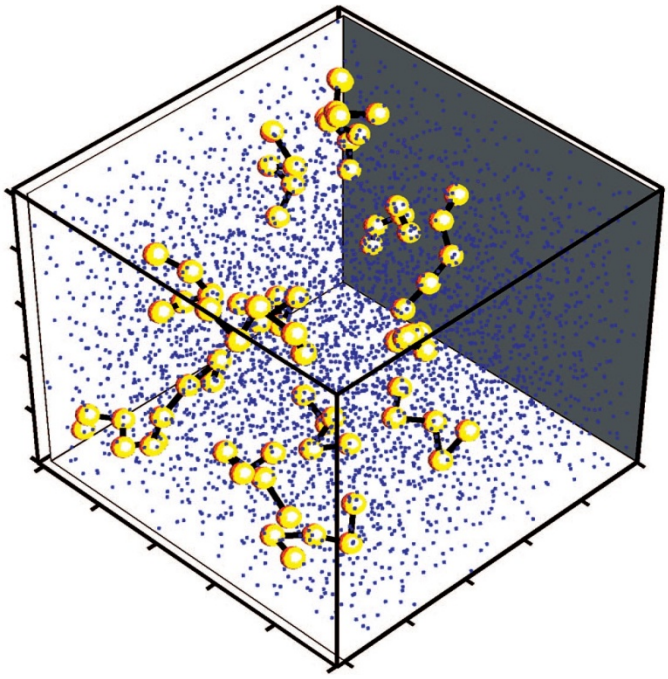

Figure 10 Polymer particles in DPD Solvent Particles (Pivkin et al., 2010).

2.6.1 Polymer Modeling Techniques

Polymers can be modeled using DPD equations where the polymer beads are subjected to conservative or repulsive, dissipative and random forces like their solvent counterparts. Along with polymer-fluid interaction, intra-polymer bead forces need to be taken into account. The repulsive force between polymer beads needs to be adjusted according to their properties. Other intra-polymer forces arise from the combination of the following (Symeonidis et al., 2005):

1. Lennard-Jones Potential

The Lennard-Jones potential provides a shifted potential to each polymer beads to avoid numerical instability. The potential is applicable to all pair particles within the

cutoff radius, $r_{ij} < r_C$. Lennard-Jones is used to prevent over-lapping. This is used instead of the soft-repulsive forces of DPD.

2. Hookean and Fraenkel

This force is a result of pairwise potential with an equilibrium distance between the beads, r_{EQ} (Symeonidis et al., 2005). The forces are calculated such that an attractive bond is created when $| r_i - r_{i-1} | > r_{EQ}$ and repulsive bond when $| r_i - r_{i-1} | < r_{EQ}$.

3. FENE

The FENE spring force is a nonlinear elastic force which keeps the bead within an equilibrium or maximum distance, r_{max}. When the distance is greater than r_{max}, the beads are pulled closer to each other along the line between the beads. K is the spring constant.

$$U_{FENE} = -\frac{\kappa}{2} r_{max}^2 log\left[1 - \frac{|r_i - r_{i-1}|^2}{r_{max}^2}\right] \quad (39)$$

4. Marko-Siggia Worm-Like Chains (WLC)

Experimental and theoretical studies showed similarities between worm-like chain (WLC) forces and DNA molecules (Perkins et al., 1995). WLC models can be used to mimic DNA movement within a flow. Similar to the FENE spring, the Marko-Siggia force expression creates a spring force that will pull the beads back closer together when the maximum length of the segment between the beads is reached. The WLC force is as follows (Fan et al., 2006):

$$F_{ij}^{S} = -\frac{k_B T}{4\lambda_p^{eff}}\left[\left(1-\frac{r_{ij}}{l}\right)^{-2}+\frac{4r_{ij}}{l}-1\right]\hat{\mathbf{r}}_{ij} \quad 40$$

where l is the maximum length of the segment between the beads and λ_p^{eff} is the effective persistence length of the polymer spring chain. The persistence length is the maximum length when the chain will be elastic. The length of the segment is established to be greater than the persistence length such that the chain will be semi-elastic or stiff. If the number of beads is N_{pb} and L is the total length of the DNA strand, then $l = L/(Npb - 1)$ (Fan et al., 2006). (Bustamante, Marko, Siggia, & Smith, 1994) stated that the persistence length of a DNA is $\lambda_p \sim 0.053\mu m$. When the beads are modeled with this length, there was an increase in the molecular flexibility since there was no bending momentum. This flexibility was resolved by increasing the persistence length to $\lambda_p^{eff} \sim 0.061\mu m$ for 40 beads or $\lambda_p^{eff} \sim 0.07\mu m$ for 80 beads in a strand with L=67.2 μm (Larson et al., 1997). In this study we have used worm-like chains to model the DNA strands.

The stress tensors are given by the Irving-Kirkwood model (Fan et al., 2006) as:

$$\begin{aligned} S \quad &= -\frac{1}{V}\left(\sum_i m u_i u_i + \frac{1}{2}\sum_i \sum_{j\neq i} r_{ij} f_{ij}\right) \\ &= -n\langle m u_i u_i + \frac{1}{2}\sum_{j\neq i} r_{ij} f_{ij}\rangle \end{aligned} \quad 41$$

where $\mathbf{u}_i = v_i - \bar{\mathbf{v}}(x)$ is a velocity difference between the bead velocity and stream velocity, $\bar{\mathbf{v}}(x)$, at position x. The ensemble average is calculated between <…>. The force f_{ij} is the result of the sum of DPD forces and the WLC spring force. The

constitutive pressure can be determined from the stress tensor, $p = -1/3trS$ (Fan et al., 2006).

The modified velocity-Verlet algorithm is also used to show polymeric interaction between particles in terms of beads joined by strings. This includes the soft repulsive force as well as hard forces.

2.7 Simulation Approach and Requirements

To carry out the present simulation, a code was programmed in Matlab including the DPD force equations and its parameters. Matlab is a high-level programming tool from Mathworks that is both interactive and versatile. The language has a multitude of in-built functions that would save time as compared to programming in C or FORTRAN. Coding in Matlab provides the ability to see the simulation results in real time when the program was being run. This was very handy at the beginning of the code production.

2.7.1 Approach to Programming

Initial positions, initial velocity and the forces for different types of particles; fluid, wall, and polymer, were initialized in separate functions within Matlab. The final Verlet algorithm was written in a script file that would call these functions when the positions, velocities and forces needed to be computed. The functions can be called within other functions as well, for example, initial position and velocity can be called within the force function to run tests.

The variables can be administered at the beginning of the script file but should be declared as a global variable when functions need the same variables as well. The variables must be unchanging to be declared globally. If a parameter is constantly changed, for instance within a script, the function would be called along with the changing variable. Each file is saved as .m file which can be read and written as a .txt file as well. The workspace which stores all the output variables can be saved as a .mat file. This file can be exported into an Excel sheet for further plotting and recording. Data stored in Excel sheets can also be imported into Matlab as .mat files.

2.7.2 Requirements

Apart from the software Matlab, one would also require a relatively high powered computer that can perform the simulation as well as other tasks simultaneously. The computer that was used for the coding had the Intel i7, which is a quad-core processor, and 4 GB of RAM. About 2GB of RAM and one of the processors will be used constantly while running Matlab. The version of Matlab that was installed is Matlab Student version R2011a. If parallel computing needs to be used, the additional processors will be employed. Parallel computing that was performed for this simulation will be discussed next.

2.7.3 Parallel Computing

The main parallel computing was performed in the Linux cluster located at Oklahoma State University and it is called Pistol Pete high performance computing. This cluster contained about 1024GB RAM and 512 cores of processors. Parallel computing can also be performed on our local machine using functions such as matlabpool, parfor and so on. Matlabpool opens workers that can perform independent jobs simultaneously. The number of workers corresponds to the number of cores in a processor, for example, a quad core processor can open an additional 4 workers along with the main Matlab program. This can only be done if the jobs are independent of each other and does not share its output.

CHAPTER III

3. RESULTS AND DISCUSSION

3.1 Initialization of the Simulation

The two-dimensional simulation region is setup on a rectangular lattice with L_X = 12 and L_Y = 30 (the unit length is r_c). The fluid, wall and DNA particles are containined within this region. The origin of the axis is located at the middle of the region. Matlab was used to write the code and to perform the DPD simulation. The following properties were used to conduct the simulation:

a) Initial particle arrangement in face centered cubic (fcc) lattice with $N=2Q^2$ particles.
b) Maxwell-Boltzmann's distribution for initial velocity.
c) Modified Velocity-Verlet algorithm.
d) Periodic boundary condition in the x-direction.
e) Frozen solid wall particles at y = - $L_Y/2$, $L_Y/2$ with a layer of no-slip condition where the particles are given a random velocity and shot back into the fluid system.
f) Cutoff radius method was used to reduce the computational cost.
g) Worm-like chains to model DNA particles.

3.1.1 Initializations

Consider N fluid particles assigned on an fcc lattice with two particles in each $Q \times Q$ unit cell where $N=2Q^2$. The density of the particles is 4 per area square. Therefore for an area of 12 × 30 or 360 unit sq., N = 1440. To have a natural number, we have chosen N = 1458. Wall particles, N_{wall}, are arranged at a higher packing density to create more repulsion at $y = -L_Y/2, L_Y/2$. If there are 400 wall particles divided into 200 particles for each wall, the density is at approximately 16.67 units at each wall. The DNA particles are initially assigned at y = 0, and the first bead is at x = - 4.5. The beads are apart from each other by a length l_{seg} = 0.7, which is larger than the persistence length of 0.053 μm and lesser than the maximum segment length, l, of 0.8075μm. Fan et al. (2006) encountered numerical inaccuracy when the maximum length exceeded l.

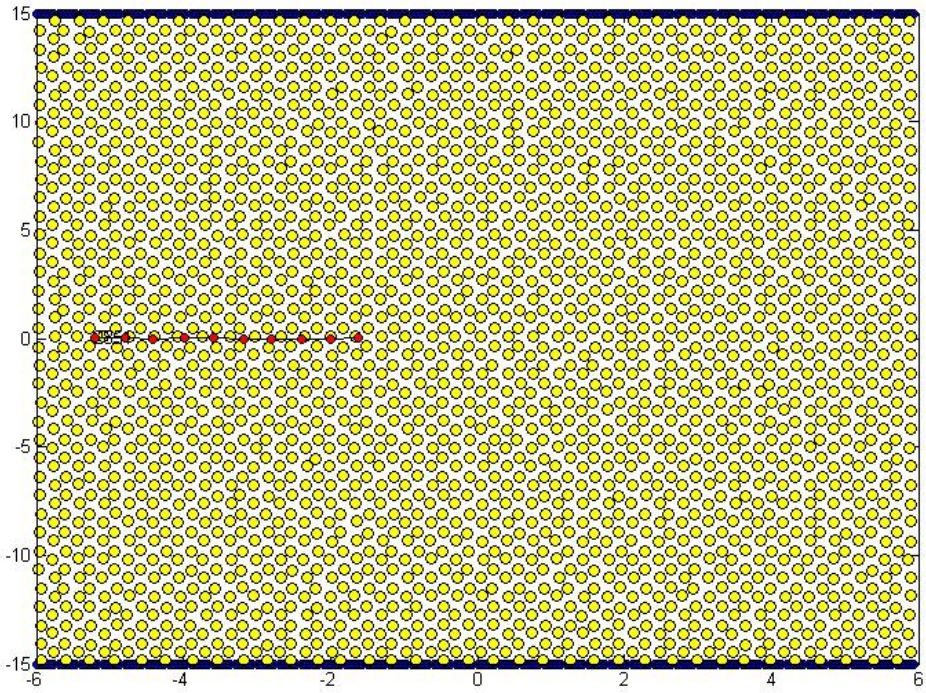

Figure 11 Initial Setup of Fluid, Wall and DNA Particles.

The computational grid is shown in Fig. 9 where the yellow is the fluid particles, the blue is the wall particles and the red is the DNA particles. The unit of length, r_c, is set to 1μm with the system dimensions as $-6 \leq x \leq 6$ and $-15 \leq y \leq 15$ similar to Fan et al. (2006).

The initial velocity is generated randomly by the Maxwell-Boltzmann's distribution with zero mean. This applied to the fluid and DNA particles whereas the wall particles were assigned a zero velocity. The velocity is curbed to be less than the maximum velocity allowed by the system. The maximum velocity is determined follows:

$$\frac{1}{2}mv_{\max}^2 = 2\frac{k_B T}{2} \tag{42}$$

or

$$v_{max} = \sqrt{\frac{2k_B T}{m}} \tag{43}$$

The total momentum of the system is set to zero.

3.1.2 Modified Boundary Conditions

There are two different boundary conditions used in this system with new modifications. The periodic boundary condition is used along the x-direction which is treated to accommodate the cutoff radius and force calculations. The periodic condition is inputted while the difference between the particles is determined. This is to define if the particles passing through the simulation region will be included in the force calculation. If a particle *i* is located at x = +5.5, then a particle *j* located at x = -5.8 will be within the cutoff radius of unity. The modifications are shown within the code in the appendix. The periodic boundary condition is used when the new position is computed at a new time

step in the main algorithm. The periodic boundary condition also allows the external force to flow through the inlet and the outlet of the channel to create a constant continuous pressure driven flow.

The second boundary condition used is the solid or frozen DPD particles at $y = -L_Y/2, L_Y/2$. Wall particles have a higher density creating more repulsive force against the fluid particle in the system. However, the soft repulsion of DPD particles would not prevent the fluid particles from penetrating through the solid wall particles, which is physically improbable. Unfortunately, higher density of the wall would create unrealistic density fluctuations in the simulation.

Therefore we have introduced the Maxwellian reflection of particles within a no-slip boundary region next to the wall. The reflection condition is administered to the particles entering into the region which has a thickness of 0.5% of the height of the channel as shown in Fig. 10. For this simulation $L_Y = 30$ and the no-slip boundary region thickness is 0.15 unit. Therefore, the no-slip region was enforced between 14.85 to 15 units and between −14.85 to −15 units. The new velocity assigned to the particles entering the region is $v_i = v_R + n\left(\sqrt{(n \cdot v_R)^2} - n \cdot v_R\right)$ where v_R is the random velocity generated by the Maxwell-Boltzmann's distribution, n is the normal vector from the wall pointing towards the fluid system where n=−1 for the top wall and n=+1 for the bottom wall. A few modifications were added along with the new velocity. New positions were also assigned to the particles for the next time step. The particles will be moved to the boundary of the no-slip region at y = 14.85 or −14.85 depending on the top or bottom wall, respectively, with a velocity vector pointing towards the center of the channel.

Forces were set to zero as the velocity was already pre-determined. This guaranteed bounce-back particles for those that reached the wall which was not penetrated.

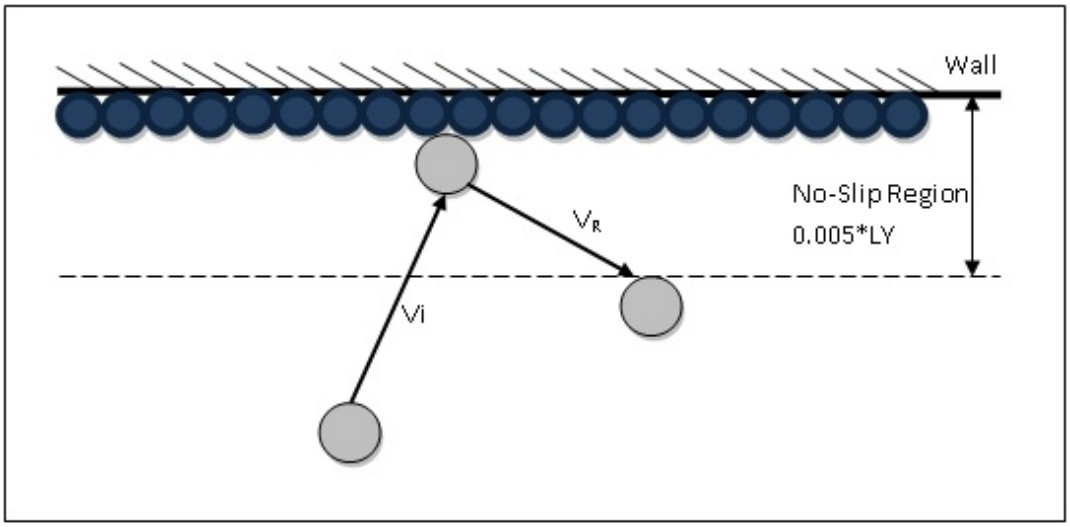

Figure 12 No-Slip Region near the Top Wall.

This method removed the need for multiple layers of wall particles to prevent the penetration of the wall by the fluid particles. The no-slip region also guarantees zero-velocity at the walls when simulating a Poiseuille flow in a channel. The particles by the wall region would not linger by the wall, creating a more practical application of the flow's physical properties.

3.1.3 Particle Forces

The force functions are needed to update the new position and new velocity of a particle at the next time step. The forces are called as each particle *i* is being computed. The force calculations are stored in three different functions. The functions are set separately as the repulsion strength for the conservation DPD force is different depending

on the type of particle interaction. When the fluid particles are in consideration, the fluid force function contains the interactive forces between fluid-fluid, fluid-wall, and fluid-polymer. When the polymer particle position and velocity is determined, the forces that contain the polymer-fluid, polymer-wall and polymer-polymer interactions are called. The spring force is also added along with the polymer DPD forces as the bead-bead interaction changes in a sheared flow.

For the spring forces, the interactions of the beads on either side of the polymer particle in question are considered. The first and the last beads will only have one bead-bead interaction, with beads *i+1* and *i-1,* respectively.

As the force is being calculated for a fluid particle, i, the force function first calculates the relative position of the particle with respect to all the other particles, in the order of fluid, wall and polymer. If the other particle, *j*, is within the cutoff radius, the force is calculated. Similarly, the polymer particles go through the process.

3.1.4 Integration

The main code contains Newton's laws for time evolving position and velocity in terms of the modified velocity-Verlet algorithm. The initial position and velocity is called into the script along with the initial force calculations. Each particle, *i,* from 1 to N, is assigned to the new trajectory after the forces are called. The positions are corrected according to the periodic boundary condition as well as the no-slip condition at the walls. The no-slip condition is enforced at the beginning of the script where the new position and predicted velocity is calculated. This would prevent the particles from bouncing back towards the wall. The no-slip is also inputted into the main particle loop to bounce-back

particles near the wall into the fluid. If the particle enters into the no-slip region with a higher velocity compared to the randomly assigned velocity, then the random velocity is used. If the original velocity is lesser than the random velocity, then the original velocity is kept while reversing the particle towards the center of the channel. This modified boundary condition is applied to both fluid and DNA particles.

3.2 Simulation Parameters

The DPD fluid and DNA parameters are the physical properties of the particles. They are listed in Table 1 and Table 2, respectively. The algorithm parameters are used to create the simulation in the mesoscopic scale and are listed in Table 3.

Table 1 DPD Fluid Parameters.

DPD Fluid Particle Parameters		
Name	**Nomenclature**	**Value**
Mass	M	1
cutoff radius	rc	1
Exponent	S	2
Fluid-fluid repulsive strength	aff	18.75
Wall-wall repulsive strength	aww	5
Fluid-wall repulsive strength	afw	9.682
Density	P	4
Verlet parameter	Λ	0.65
Random force parameter	Σ	3
Dissipative force parameter	Γ	4.5
Field force	G	0.02
Energy conservation	kBT	1
Maximum Velocity	Velmxd	1.414

Table 2 DPD DNA Parameters.

DPD DNA Particle Parameters			
Name	**Nomenclature**	**Value**	**Unit**
Effective persistence length	Λeff	0.053	µm
Energy conservation	kBTp	1	
Total DNA strand Length	L	67.2	µm
Maximum DNA segment Length	Lseg	0.808	
Mass Density	Mp	0.25	g/cm3
Viscosity	Viscp	2.588	kg/µms
Fluid-polymer repulsive strengthafp		16.5	
Polymer-polymer repulsive strength |i-j|>4	App	2	
Polymer-polymer repulsive strength |i-j|<4	App	0	
Fluid-wall repulsive strength	Apw	3.162	

Table 3 Algorithm Parameters.

Algorithm Parameters		
Name	**Nomenclature**	**Value**
Fluid particles	N	1458
Length of channel	LX	12
Height of channel	LY	30
No-slip Region	Rcw	0.15
Time step	Delt	0.02
Initial time	Ti	0
Final Time	Tf	120
Number of cells (x,y)	Q	27
Length of unit cell x direction	Nx	0.444
Length of unit cell y direction	Ny	1.111
Wall particles	Nwall	400
Wall density	Nwall	16.667
DNA particles	Np	1
DNA beads	Npb	10
Total DNA beads	Nptot	10
Total DPD particles	Ntot	1868
No-slip Region Top boundary	BC1	14.85
No-slip Region Bottom	BC2	-14.85
Bins	Tbins	35

3.3 DPD Fluid Flow in a Channel

A pressure driven flow is simulated through a two-dimensional channel with lengths, L_X=12 and L_Y=30 (in r_c units). The channel has two solid boundary walls at y= $L_Y/2$ and -$L_Y/2$ with the modified boundary conditions. The expected flow through the channel is Poiseuille flow as DPD follows Navier Stokes equations. The channel flow is investigated at different test conditions.

As the simulations were conducted, the particles velocities were averaged in spatial bins and the computed velocity profile was compared to the theoretical velocity profile of a Poiseuille flow between parallel plates, given by:

$$\frac{u}{\bar{u}} = \frac{3}{2}\left[1 - \left(\frac{y}{h}\right)^2\right] \quad 44$$

where $\bar{u}$ is the average velocity, y is the spatial coordinate in the vertical direction and h is the half-height of the channel (in the y direction). In this present study, the height of the channel was divided into several bins, including two bins for the lower and upper no-slip boundary regions. The rest of the bins were equally divided. The velocities of particles are accumulated in each bin according to their y-position for every 1000 timestep. They are summed in each bin and divided by the number of particles collected as shown in Figure 13. The average velocities did not include the ones calculated during the first few timestep since the velocity fluctuates due to the randomly assigned velocity. Once the channel flow starts forming, the velocity is collected.

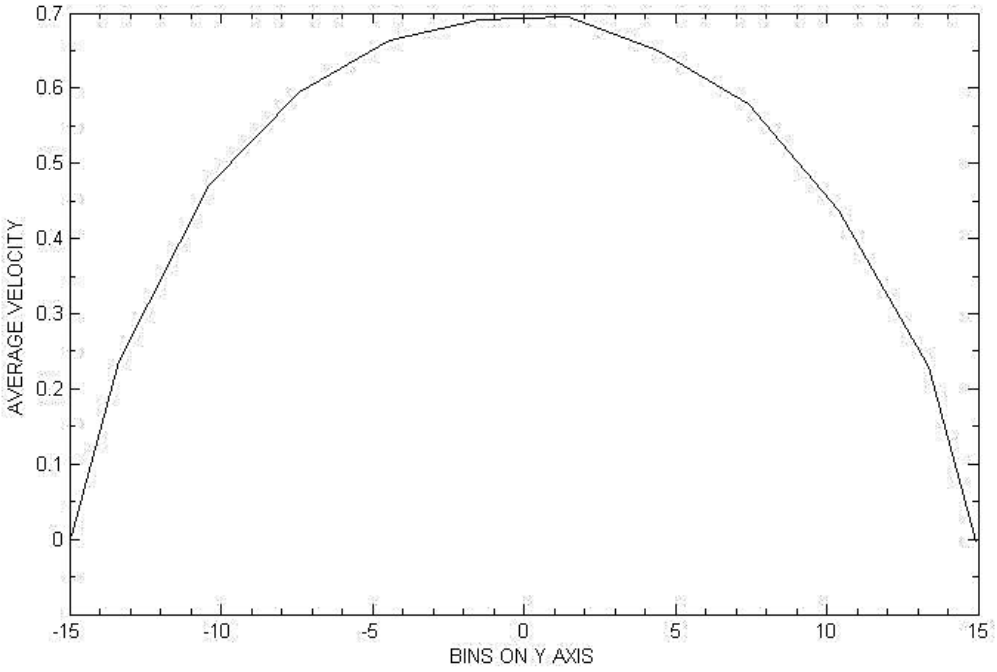

Figure 13 Average velocities per Bin with the Modified Parameter s.

3.3.1 Simulating Channel Flow using Standard DPD

The first test performed was simulating the two-dimensional flow through a channel (Poiseuille flow) using standard DPD parameters, where r_c=1 and s=2, for a number density n= 4. With these parameters, the viscosity of the simulated fluid is very low. For such a low viscous fluid, one can expect a flat velocity profile. Initially a random velocity distribution is assigned to the flow field. A fully developed flow is acquired at t=180. Due to the changes in the forces (the change in velocity is determined by forces as shown before) the velocity field develops an almost top hat velocity profile with an average value of 1.2 approximately.

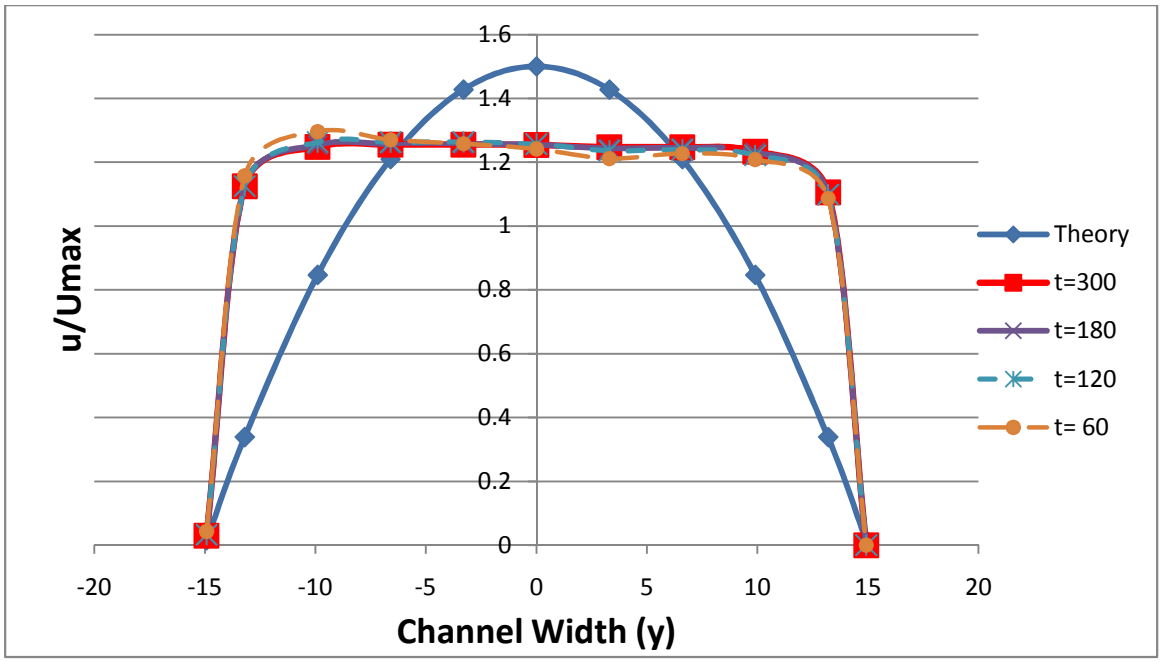

Figure 14 Poiseuille Flow using Standard DPD Fluid.

3.3.2 The Effect of the 's' Parameter

The parameter *s* is the exponent of the weighing function used to determine the dissipative and random forces for DPD. Modifying the s parameter changes the interaction between the fluid particles. It would be then possible to adhere to the physical property of a true fluid. Due to the soft repulsive force between the particles, DPD simulations are usually associated with a low viscous fluid and low Schmidt number flow.

The dimensionless number can be increased by enhancing particle interaction which is done by increasing the cutoff radius. Unfortunately, increasing the cutoff radius escalates the computational cost drastically. Alternatively, the exponent *s* can be

modified with a minimal rise in the computational cost. Schmidt number increases 10% when *s* is modified and viscosity increases by 36%.

The simulation of Poiseuille flow with the modified parameter s is shown in Figure 15. The Poiseuille flow is attained by plotting the ratio of velocity over the maximum velocity against the y axis or the channel width. The maximum velocity is a function of viscosity. This can be illustrated by Figure 14 and Figure 15 where the non-dimensional velocity profiles change with varying parameter *s* and, thus, they are a function of viscosity. The standard DPD has s=2. When s=0.5, the computational velocity profile agrees qualitatively with the theoretical Poiseuille flow equation for flow between two parallel plates. The agreement is not complete, however. This could be due to the altered no-slip boundary conditions with the randomly assigned velocity and the displacement of the particles at the boundary. However, the flow shows symmetry, as expected, and the no-slip conditions are clearly enforced at the wall boundaries. The fully developed velocity profile is acquired at t=180 and remains the same till t=300. The average velocity of the channel is 0.8645. The approximate length it would take to reach a fully developed flow will be $155.61r_c$.

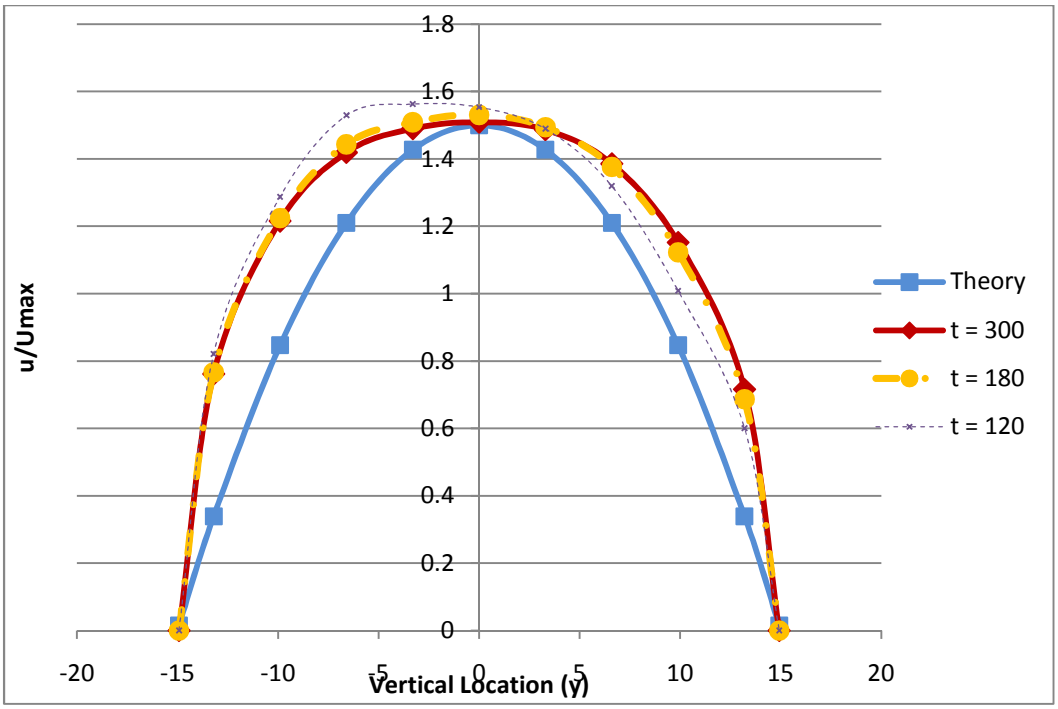

Figure 15 Poiseuille flow with Modified DPD Parameter 's'.

3.3.3 The Effect of the Channel Size

To check whether the simulation is grid-independent, a larger region was used in the simulation with L_X=60, L_Y=30 and N=7200 to keep the number density at 4. The flow profile is flat similar to the previous grid with L_X=12 and L_Y=30. The velocity profile tends to reach an average of approximately 1.2 fairly quickly, i.e. at t=60. The simulation is grid independent and will provide the same results for different sized grids.

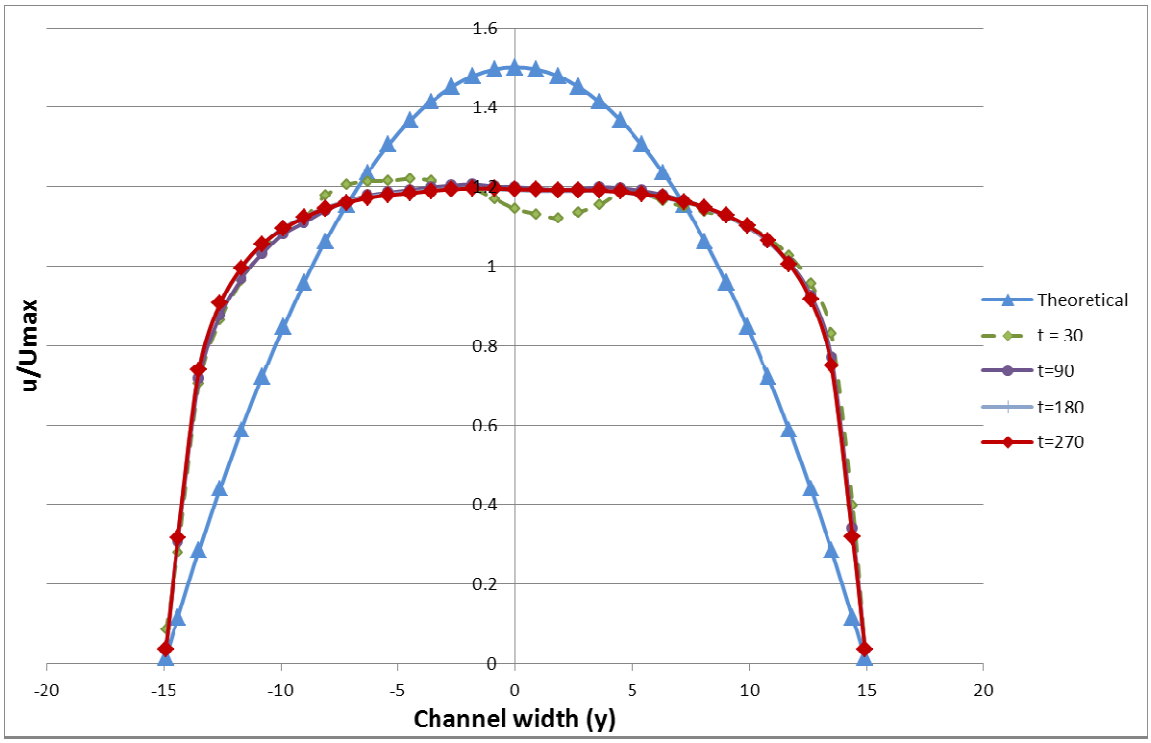

Figure 16 Poiseuille Flow with Larger Simulation Region.

When channel flows with smaller channel heights (e.g. L_X=12 and L_Y=3 or 6) were investigated, the fluid particles segregated into lines with spacing dependent on the cutoff radius as shown in Figure 17 and Figure 18. The separation distance between the lines was reduced as the cutoff radius was decreased. This phenomenon is not fully understood and it needs further study. It may indicate that the DPD equations need modifications to simulate nanoscale systems.

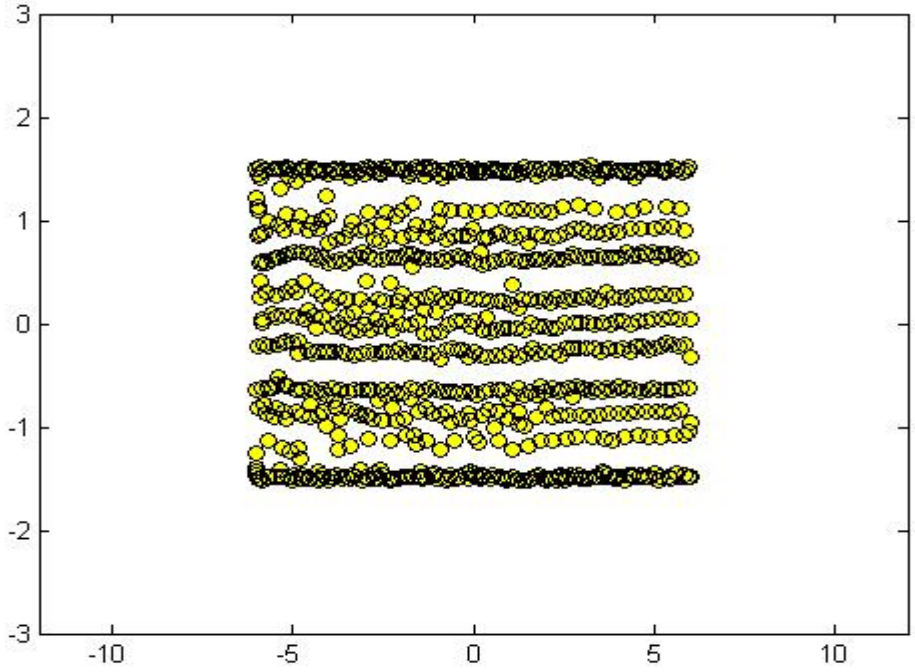

Figure 17 Simulation of System with LY=3.

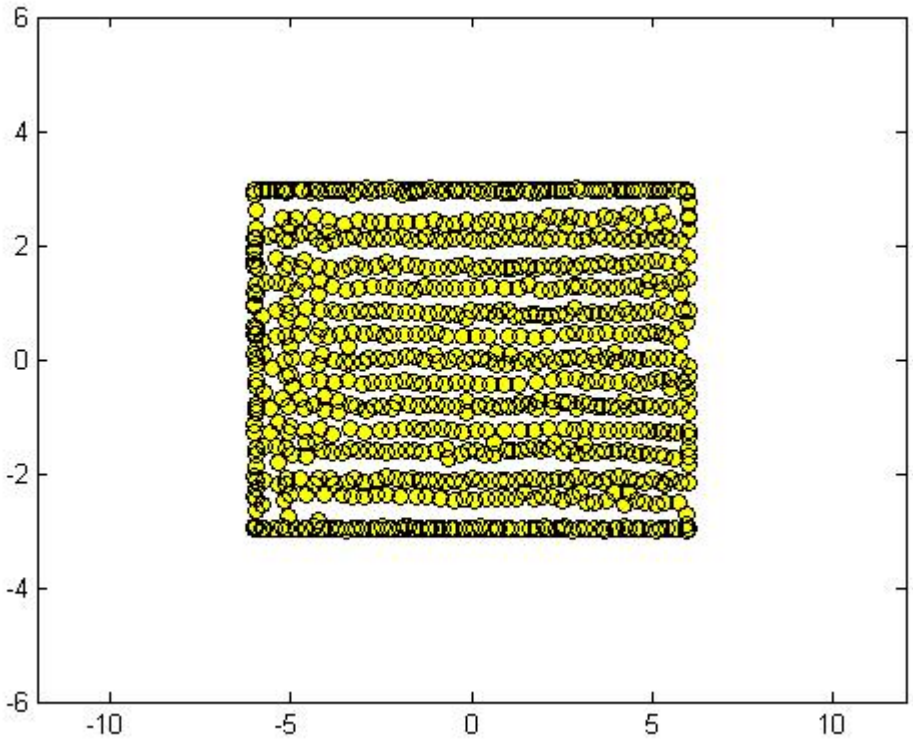

Figure 18 Simulation of System with LY=6.

3.3.4 The Effect of the Number Density

A simulation with number density of 20 is shown in Fig. 17. The velocity profile tends to get closer to the theoretical Poiseuille flow as the number density of the system is increased. The increase in the density beyond a certain limit will change the physical property of the flow; the system may not contain fluid particles at that limit and may instead act as solid particles. The particles may also be incompressible after a certain number density limit. These effects of increasing number density within DPD and the compressibility of the flow deserves a detailed study.

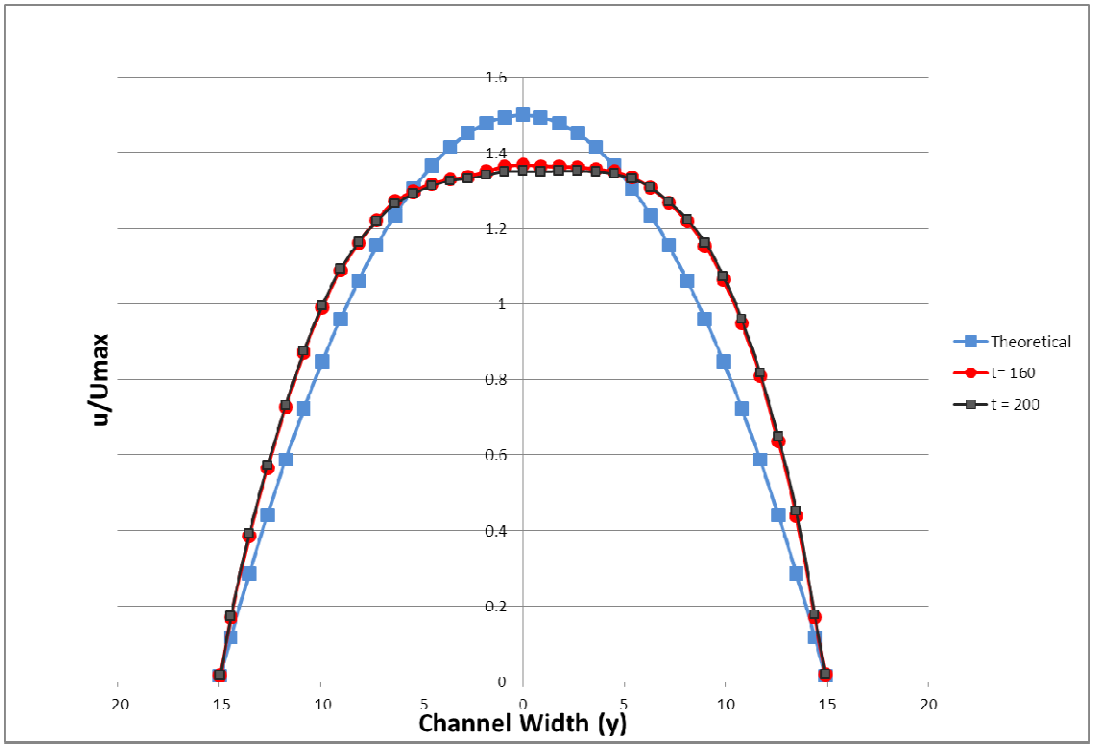

Figure 19 Poiseuille Flow with Number Density =20.

3.4 Simulating a Single DNA Strand in DPD Flow

DNA strands were placed at different locations in the channel where they experienced different values of shear stress gradients to observe their migration and physical properties as they travel through the pressure-driven channel. DNA chains tend to stretch, fold and tumble as they move through the channel. These properties are relevant to understanding the flow of DNA in lab-on-a-chip devices. The present tests were conducted with one strand at a time but with different number of beads and at different channel locations.

The DNA strand tends to extend or fold at different positions within the channel. These extensions relax after a time period and remain in a constant state after the flow profile is developed. The time it takes for the strand to reach a particular extension is related to the strand relaxation time. The relaxation time, τ, can be computed from observing the extension of the strand, x, as function of time, t, and plotting

$$x = x_0 + x_i \exp(-t/\tau) \quad (45)$$

where x_0 is the extension at equilibrium and x_i is the maximum extension of the strand.

3.4.1 DNA Migration in a Channel Flow

Initially there are 10 DNA beads on a strand placed at different y positions along the height of the channel. The positions are -13, -10 and -5. This test was conducted to see if the DNA strand would migrate to the centerline of the microchannel. The simulation is performed using the standard DPD parameters and the viscosity of the fluid

particles is not corrected. Therefore, the velocity profile is almost a flat profile. Due to the no-slip boundary conditions at the wall the flow field possesses larger shear stress gradient near the wall compared to the region near the centerline. The migration of a DNA strand with 10 beads from different vertical locations along the channel height (Y= −13, −10 and −5) is plotted Fig. 18.

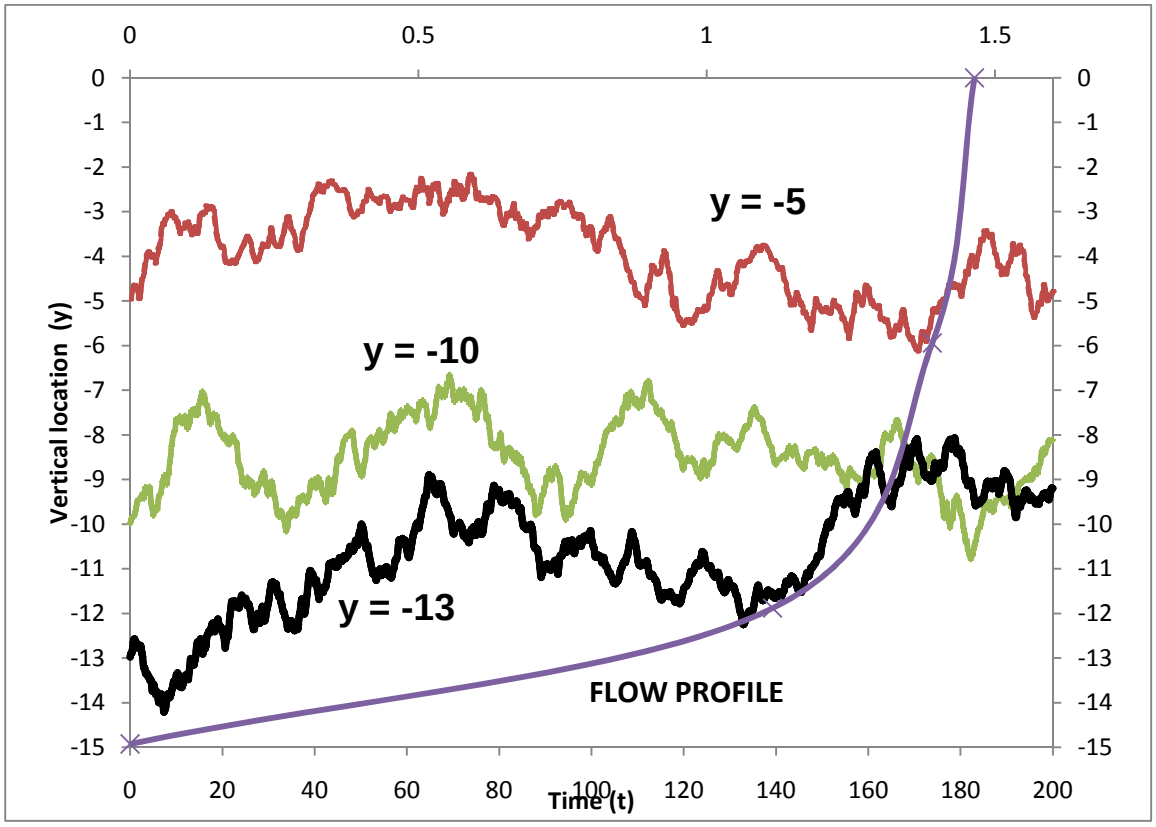

Figure 20 Migration of DNA Strands from Different Positions at s=2.

The strand inserted initially at position y = −13 experiences larger shear stress gradient compared to the two other strands. As a result, the strand migrates towards the centerline of the channel faster than the two strands starting their journey from y= −10, −5. The strand at y = -10 moves closer to the centerline but at a much slower pace, due to the

smaller level of the local shear stress gradient in this region. This can be seen from Figure 20 and Figure 21 where the strand has only moved from y=-10 to y=-8 in t=200 units, but moves till y=0 at t=984 units. This is because the velocity profile is flatter towards the center of the channel and the strands need more time (i.e. longer channel) to migrate to the centerline.

3.4.2 Effect of the Number of Beads on DNA Migration

Three strands with different number of beads are inserted into a pressure-driven flow at y = -10, near the bottom wall of the channel. The strands have 5, 10 and 20 beads respectively. Similar to the previous test, the flow has a flat profile due to the low viscosity and less particle interactions for s=2. Figure 21 shows the migration of the simulated strand for s=2. The migration is more prominent for the strand with 10 and 20 beads. The strand with 5 beads seems to oscillate along the initial position of y = -10 and then slowly reaches y = -8 at t=1200. The 20 beaded strand appears to have migrated much faster than the other two strands and reaches the centerline by an approximate time, t = 899. The strand with 10 beads reaches the centerline later at t=984. The strands tend to migrate towards the centerline due to the presence of a varying shear stress gradient. If the length of the channel is long enough the strand finishes its journey at the centerline. Longer channels (i.e. longer times) are needed for strands with less number of beads to reach the channel centerline.

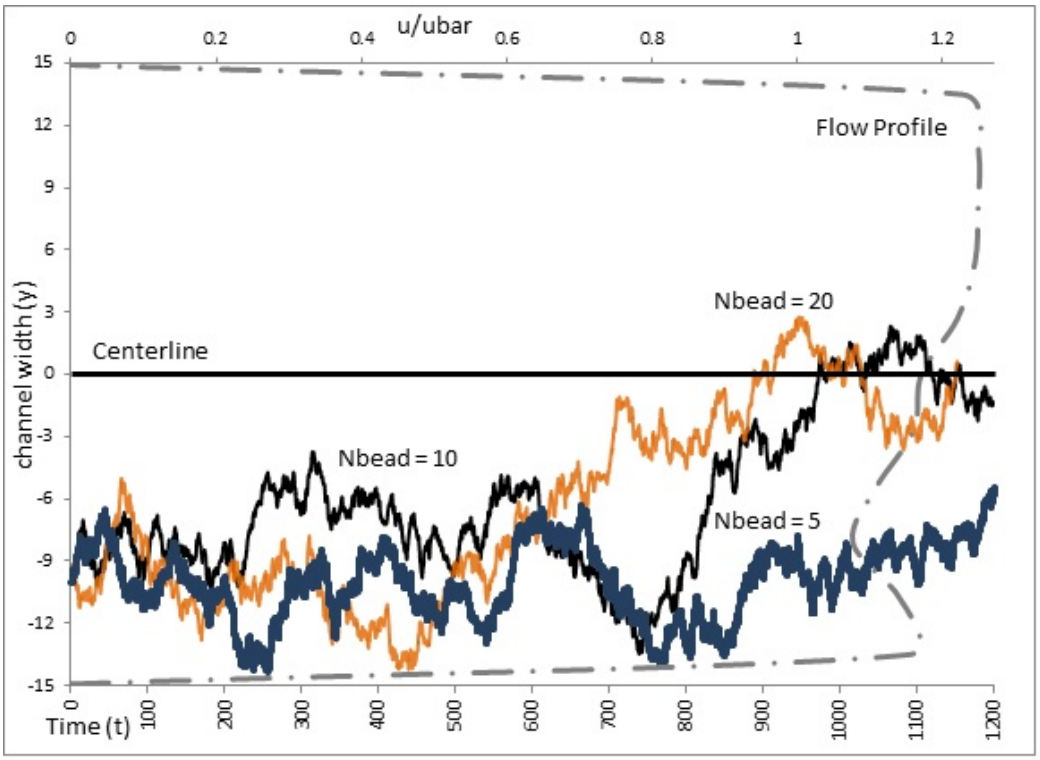

Figure 21 Migration of DNA Strands with Different Number of Beads at s=2.

Figure 22 on the other hand shows a flow with a modified s=1.5. This increases the viscosity and particle interaction. The strands tends to oscillate along the initial position of y=-10 between y=-9 and -12. Figure 23 illustrates the DNA migration at s=1. The strand with 5 beads reached the centerline as compared to the strands with 10 and 20 beads that finished closer to their original positions on the y axis. Figure 24 illustrates the travel of the strands which uses the modified parameter s=0.5. The velocity profile in this case is more like a theoretical Poiseuille flow. The strands tends to linger at their original positions, perhaps due to a constant shear stress gradient compared to the varying shear stress gradient in the case of s=2. The strands with 5, 10 and 20 beads oscillates at its original position y=-10.

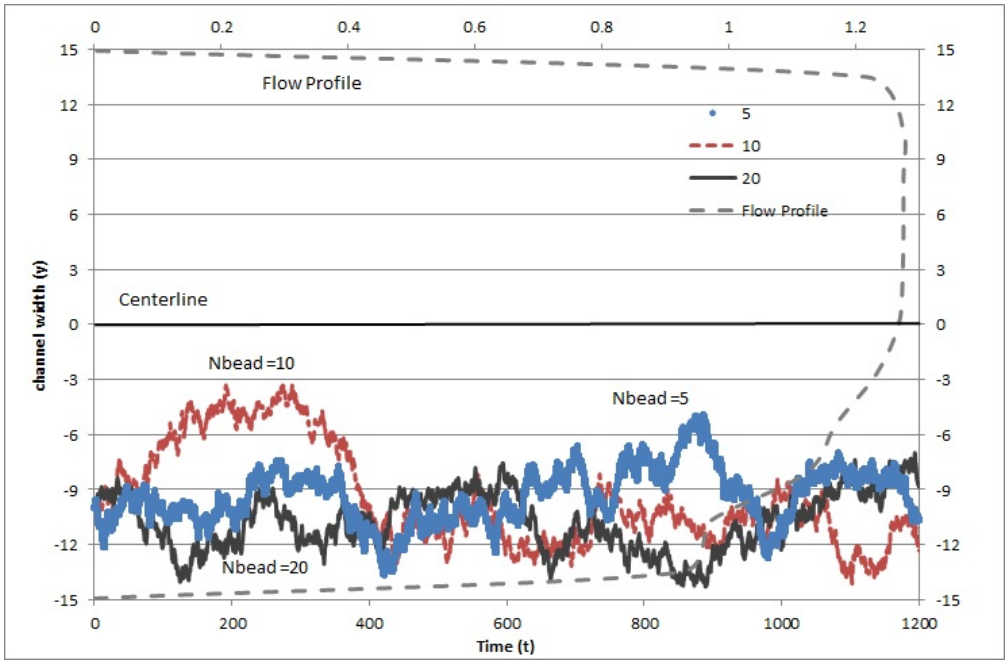

Figure 22 Migration of DNA Strands with Different Number of Beads at s=1.5.

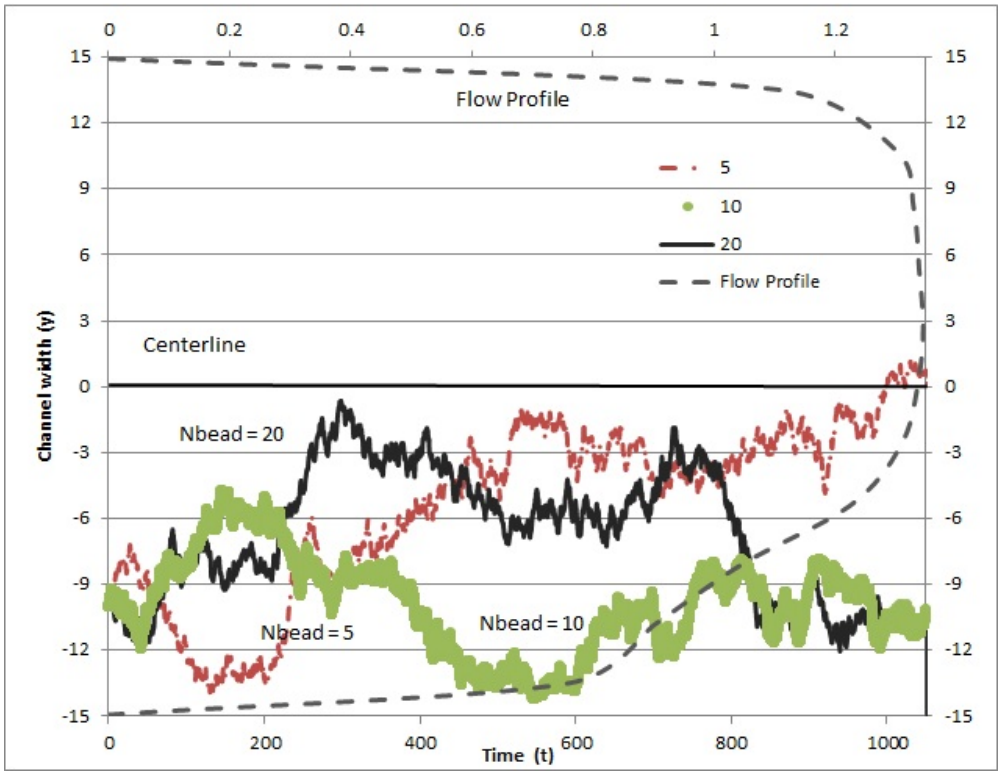

Figure 23 Migration of DNA Strands with Different Number of Beads at s=1.

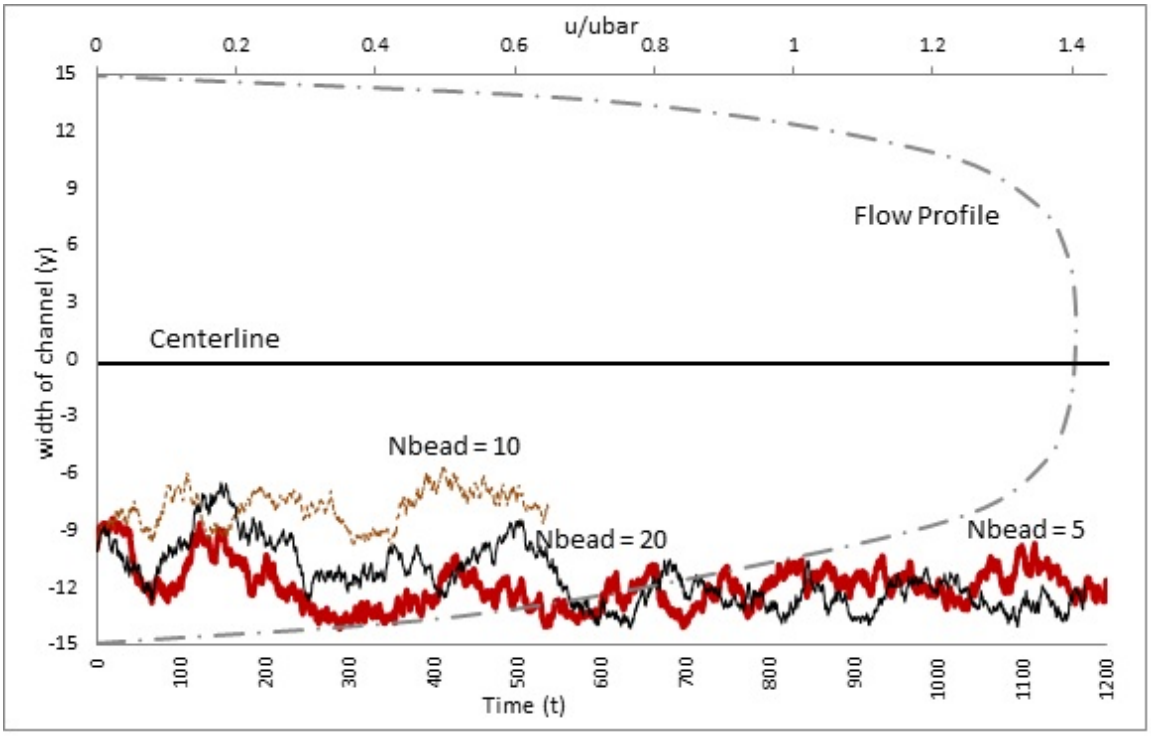

Figure 24 Migration of DNA Strands with Different Number of Beads at s=0.5.

As the plots vary from s=2 to s=0.5, we can clearly see the velocity profile approaching the shape of the velocity profile of the Poiseuille flow. The DNA particles seems to move towards the centerline when s =1 and 2 as compared to the steady oscillation along the initial inserting location when s=0.5 and 1.5. The effect of modifying the parameter *s* on the physical properties the DNA or polymer particles is not clear. To maintain accurate viscous behavior for DNA particles one may need other values of *s* or different cutoff radius r_c.

Figure 25, Figure 26 and Figure 27 compares the migration of the beads for different values of the parameter *s*. As mentioned before, longer beads tend to travel more towards the centerline as s=2; the value for a standard DPD fluid. A strand with 20 beads

tends to migrate faster than the shorter strands. This is observed in Figure 23 where the 20-beaded strand attempts to move towards the centerline at s=1. Longer strands must be utilized to understand the relation between the weighting functions and the DNA particles. Further analyses are necessary to understand if the weighting function can be changed to attain accurate Schmidt number and the physical properties of DNA strands. The simulation would need a longer channel or more computational time to average the migration patterns of these long strands.

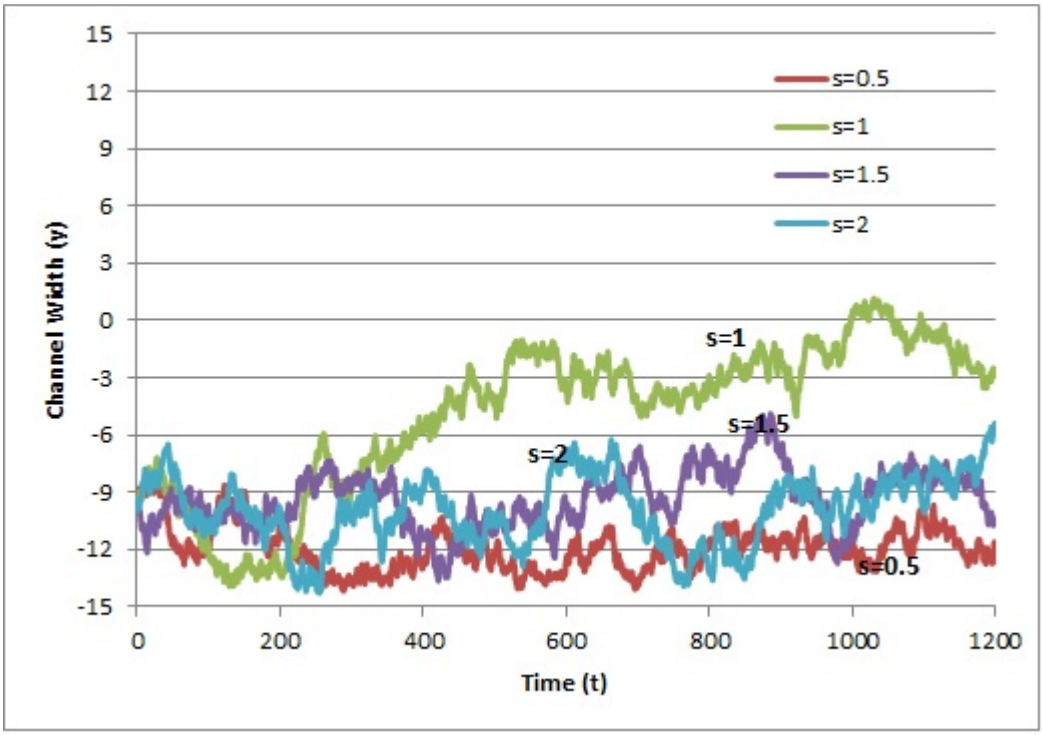

Figure 25 Migration of Strand with Nbead = 5 for Varying s.

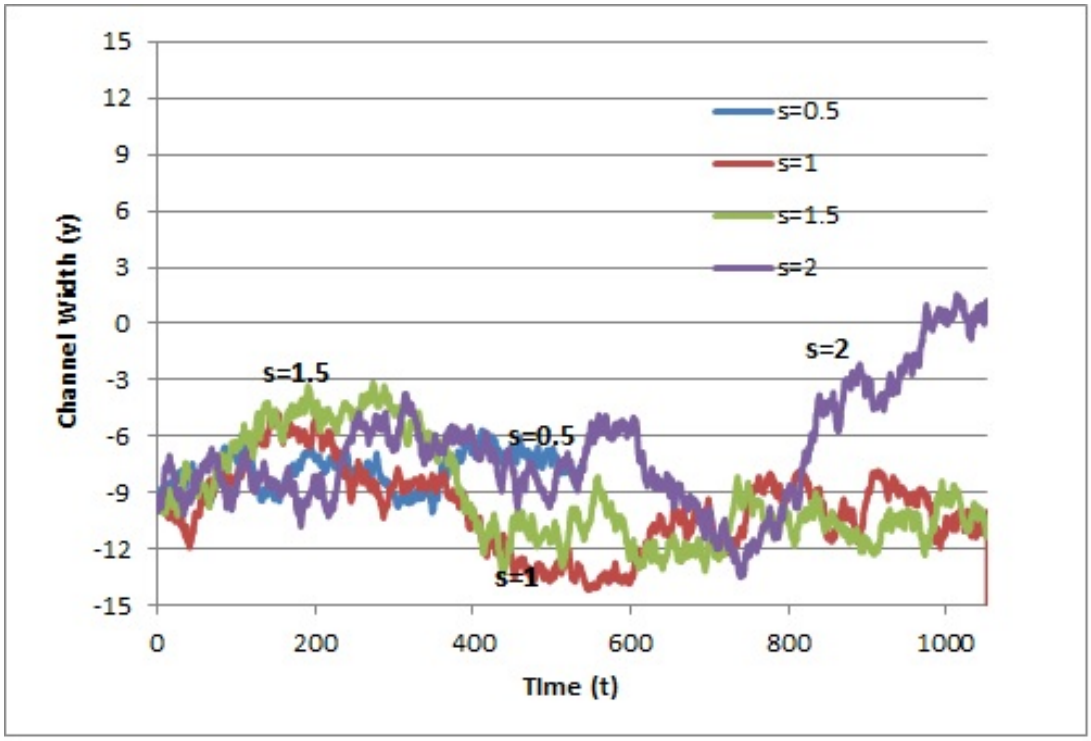

Figure 26 Migration of Strands with Nbead = 10 with Varying s.

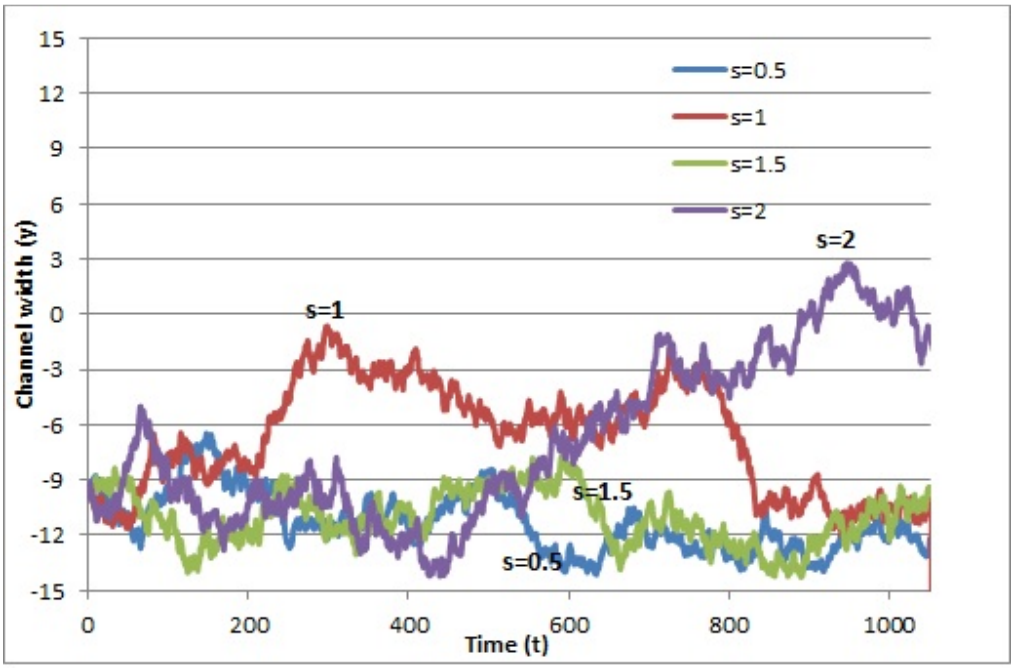

Figure 27 Migration of Strand with Nbead = 20 for Varying s.

3.4.3 Extension with Respect to Time

As the DNA strand migrates through the channel, it stretches and folds due to the shear stress produced by the velocity profile. Figure 28 and Figure 29 shows such a behavior for a strand at two different locations, y = 0 and y = -10, respectively. The velocity profile is flat due to the low viscosity and Schmidt number. There the strand at y=0 does not have a high shear stress acting on it compared to the strand at y=-10. Due to the difference of shear, the DNA strand at y=0 folds and tumbles as it flows through the channel. The strand at y=-10 stretches out as it moves across the channel. The extension rate can be measured with respect to time to compute the relaxation time.

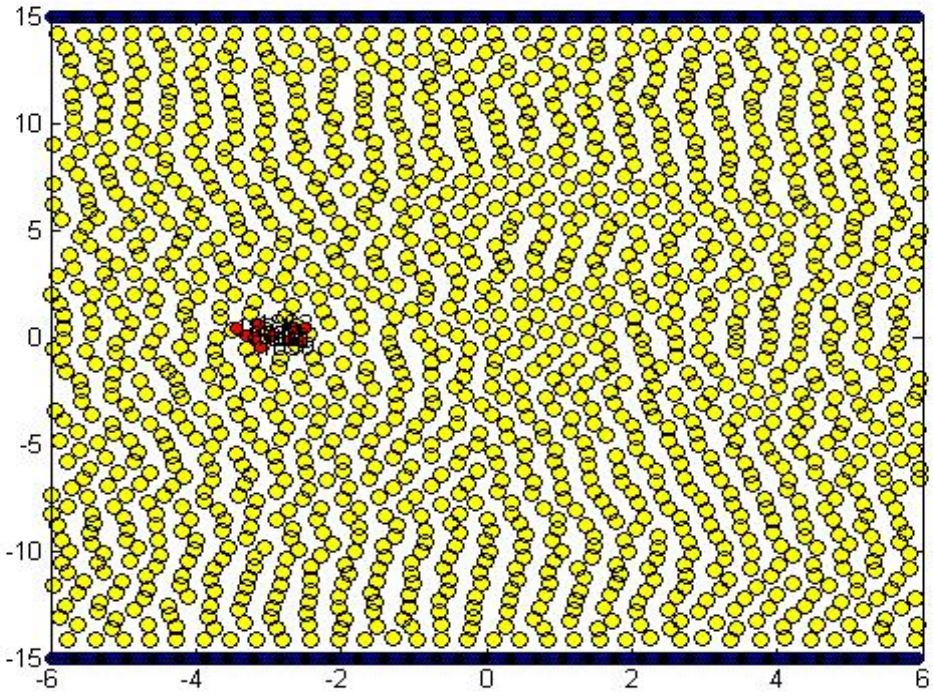

Figure 28 DNA Folding at y=0.

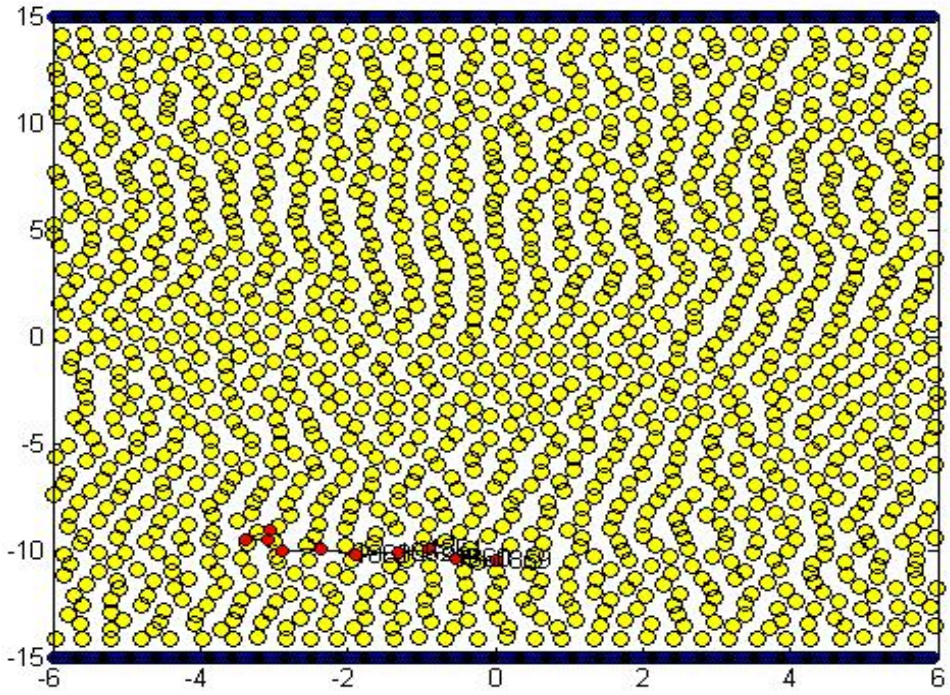

Figure 29 DNA Stretching at y= -10.

Next we are simulating a strand with 40 beads in a channel. The simulation is conducted with both zero external force and a high external force. The results are then compared to the simulation data provided by Chun Cheng et al. (2008). The extension rate of the DNA strand is measured by finding the distance between two beads of the strand located at either end of the strand at time t. This may not necessarily be the first and last bead of the chain as the beads tend to fold and tumble. The present code goes through the position of each bead at every timestep and finds the position of the downstream-end and upstream-end of the chain. The fractional extension is computed as the ratio between the extension length, l_{ext}, and the total length of the DNA strand, L (Chun Cheng, Feng, Qian Qian, & Xiang Dong, 2008) as follows:

$$fractional\ extension, \mathrm{l}_{\mathrm{fract}} = \mathrm{l}_{\mathrm{ext}}/\mathrm{L} \quad (46)$$

The fractional extension is then plotted against the time and the relaxation time is computed by fitting an exponential curve to the plot.

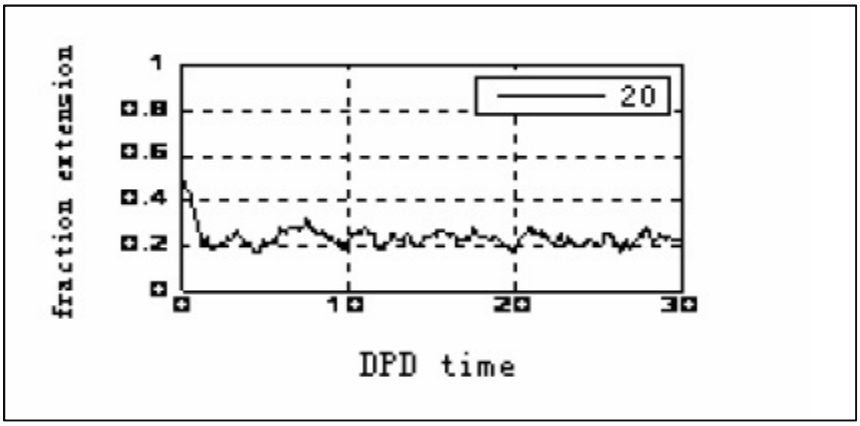

Figure 30 Fractional Extension with Respect to Time (Chun Cheng et al., 2008).

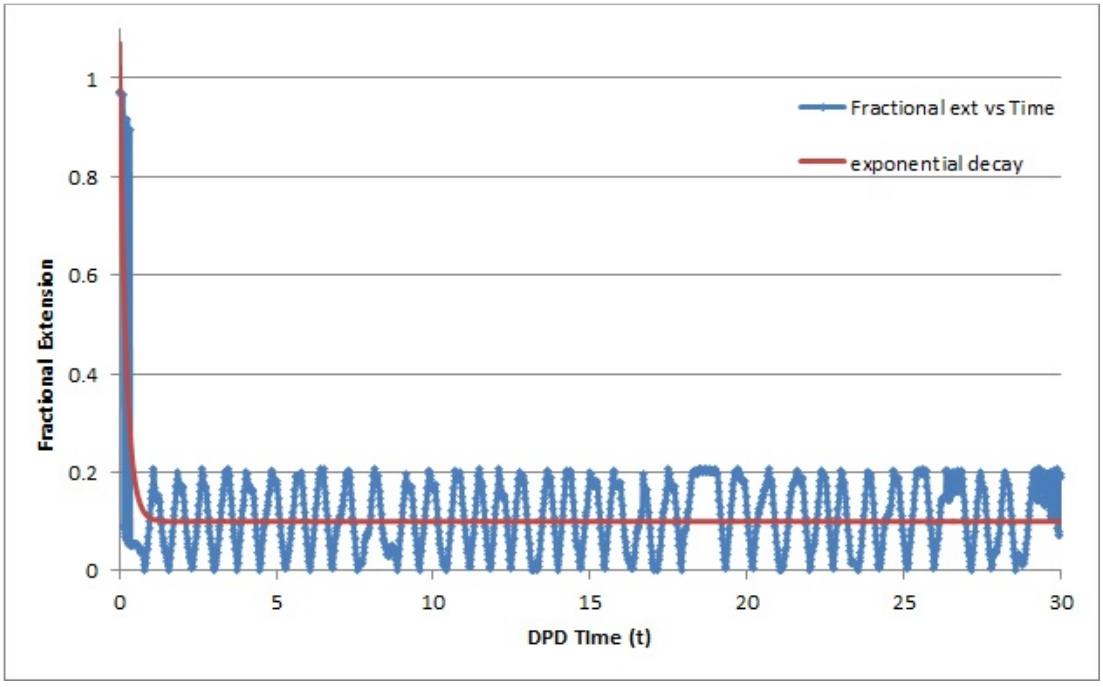

Figure 31 Fractional Extension vs. DPD Time and Exponential Decay.

Figure 28 shows the results of Chun Cheng et al. (2008) for low external force and the extension averages at 20%. In this study, we compute the extension with zero external force and the fraction extension averages over 10%.

$$\text{extension}, x = 0.1 + \exp\left(-\frac{t}{0.2}\right) \quad (47)$$

The relaxation time is calculated to be 0.2 DPD time units at zero external force for a DNA strand with 40 beads. Figure 32 shows the extension averaging at 0.1 and then oscillates about 0.05 after t=7.5. The extension is simulated at g=0.1.

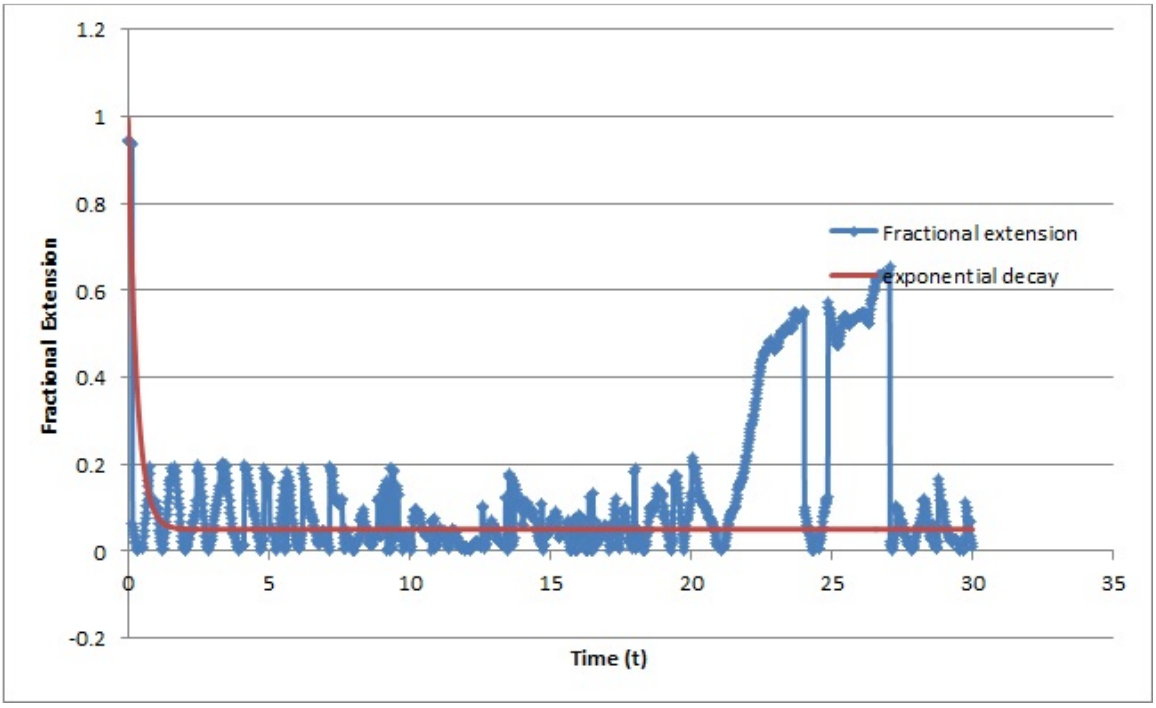

Figure 32 Fractional Extension at External Force, g=0.1.

Figure 33 is the fractional extension with an external force, g=10. The plot illustrates how the DNA strands reached its equilibrium extension at 5% of its original length much quicker with a higher external force. The peaks in the plot are due to the DNA leaving the periodic boundary condition at x=6 and continues at x=-6. However, this shows that the DNA strand would reach equilibrium much faster at a higher external force.

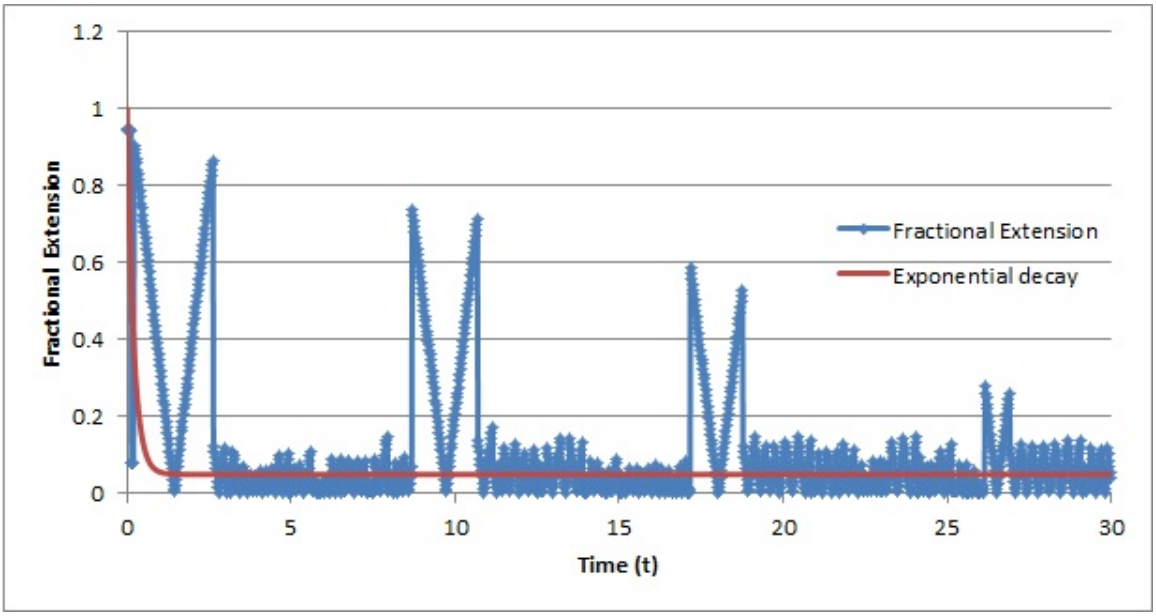

Figure 33 Fractional Extension with External Force, g=10.

3.5 Remarks and Discussion

In order to have a more realistic simulation, one shouldpay more attention to (1) matching the solvent and (2) physical properties of the DNA strands especially the charge of the strands. Fan et al. (2006) simulated and compared his findings to Perkins et al (1994) experiments. The solvent used in the experiment was buffer solution of tris-HCL,

EDTA, Tween-20 and NaCl (Perkins, Smith, & Chu, 1994). He formed a single DNA by attaching multiple λ-phage DNA molecules to create a DNA strand upto 100 µm. Fan et al. (2006) simulated that extension and was able to compare the data to Perkins et al (1994) findings.

CHAPTER IV

4. CONCLUSIONS AND RECOMMENDATIONS

4.1 Summary and Conclusions

In this research, we present DPD simulation of Poiseuille flow with new modifications to the DPD formulation and the boundary conditions. The viscosity and particle interactions issues for standard DPD can be corrected by modifying the weighing function of the dissipative and random force. This increases the viscosity of the DPD fluid particle to relate to the true fluid and keeps the computational cost to a minimum. The boundary conditions are altered such that the no-slip region prevents the fluid particles from penetrating through the solid or the 'frozen' wall particles. This also reduces multiple layering for the wall particle to increase its density. The no-slip region prevents drastic density fluctuations at the wall. The modifications have provided a valid Poiseuille flow profile for a two-dimensional system. The disagreement between the simulated flow profile and the theoretical Poiseuille flow may be due to the modified boundary condition or the changed parameter *s.* This should be tested by changing the parameter s further or by varying the cutoff radius instead. Table 5 shows that the Schmidt number is proportional to r_C^8 and therefore the interactions can be increased faster than the parameter *s*.

DNA or polymer migration is tested through the microchannel using worm-like chain. The spring forces along with the DPD forces simulate DNA strands. We do not have experimental data to compare the DNA migration through a Poiseuile flow, but since worm-like chains are assumed to be able to replicate DNA strands, we have attempted to gather physical characteristics of DNA movement through a microchannel. Longer DNA strands appear to migrate towards the centerline for both standard DPD weighting functions and modified parameter. However, more experimental data is required to gather the interaction coefficients between the DNA and solvent particles. With such data, DPD can be modified to increase the particle interaction without a rise in computational cost. The DNA strands influences the velocity profile such that there is a dip where the DNA strand is originally placed. However, the flow profile is less affected within DNA with a decreased parameter *s*. This may be due to the viscous effect of the DNA strands on the fluid particles.

The worm-like chains illustrates the conformation of DNA strands as it travels through the microchannel. One can observe the folding, coiling, tumbling and entanglements of the strand with varying external forces. The DNA strand would tumble and coil if the external force was minimal and the interactions between the particles were reduced. The strand stretches and remains relatively extended if the external force and the interaction was larger. The extension relaxes quickly with a stronger external force. As longer strands entangles, strong internal forces are created which may cause more molecular deformation. With zero external force, the strand relaxes to a set extension slowly and oscillates about 0.1% of it original length. As the external force is increased to

g=0.1, the DNA strand relaxes much faster from 0.1% to 0.5% of its maximum length. When g=10, the strand relaxes to 0.05% of its maximum length much faster.

4.2 Recommendations For Future Work

Dissipative particle dynamics has a lot of room for improvement and there has been continuous changes and additions made to this scheme. For example, Smoothed Dissipative Particle Dynamics is a modification of DPD with additional hydrodynamic forces added to the original method. Also, there is a need to understand the interaction between polymer and different types of solvent particles and experimental data for such interactions are still sought after especially for channel flow. Most of the data available are for Couette flows with tethered DNA particles. The weighing functions can be adjusted for the solvent particles, but when the DNA particles are added, the weighing function parameter needs to be altered separately. DNA simulation using DPD with modified weighting functions, both parameter s and the cutoff radius, can be improved further with the availability of computational resources and experimental data.

The computational cost of this code was relatively high. When the number of particles is increased to over 7200 for 10000 time step, it would take about 45 hours. For this system, with 1458 particles, it took 2.2 hours for 10000 time step. The only computational efficieny method used was the cutoff radius. If the neighbor list or cell list method is added to the algorithm, the code would be much faster. Also, Matlab is a useful tool to visualize the flow phenomena as the code was applied. However, programming

languages such as C or Fortran would reduce the bottlenecks and limitations of Matlab programming.

DPD is a powerful tool that can be used for a variety of computational simulation. It is less expensive that MD or MC in terms of computational cost. However, it can be improved if parallel computing is utilized. Unlike MD, DPD has lesser distributed computation as each calculation is depended on the other. If DPD is modified to compute forces separately, the cost would be reduced drastically. Also, the equation for DPD seems to be limited for microscopic to mesoscopic flows. If the system is too small, the flow field tends to segregate into sections within the system which requires further studies. Modifications for DPD forces in a nano-structure can be improved with further study and the availability of experimental data.

Finally, the morphology of the actual DNA molecule is more complicated to be simulated with Worm-like-Chain method as presented in the present work. More realisitic models would be necessary to duplicate the physical properties of DNA.

REFERENCES

Arya, G., Chang, H. C., & Maginn, E. J. (2003). Molecular simulations of Knudsen wall-slip: Effect of wall morphology. [Proceedings Paper]. *Molecular Simulation, 29*(10-11), 697-709. doi: 10.1080/0892702031000103257

Bustamante, C., Marko, J. F., Siggia, E. D., & Smith, S. (1994). Entropic elasticity of lambda-phage DNA. *Science (New York, N.Y.), 265*(5178), 1599-1600.

Chun, Kyoseok, et al. "Fabrication of Array of Hollow Microcapillaries Used for Injection of Genetic Materials into Animal/Plant Cells." *Japanese Journal of Applied Physics* 38.2-3A (1999): 279-81. Print.

Chun Cheng, Z., Feng, J., Qian Qian, C., & Xiang Dong, S. (2008, 6-9 Jan. 2008). *Simulating stretching dynamics of DNA with dissipative particle dynamics.* Paper presented at the Nano/Micro Engineered and Molecular Systems, 2008. NEMS 2008. 3rd IEEE International Conference on.

Fan, X., Phan-Thien, N., Chen, S., Wu, X., & Ng, T. Y. (2006). Simulating flow of DNA suspension using dissipative particle dynamics. *Physics of Fluids, 18*(6), 063102.

Fan, X., Phan-Thien, N., Yong, N. T., Wu, X., & Xu, D. (2003). Microchannel flow of a macromolecular suspension. *Physics of Fluids, 15*(1), 11-21. doi: 10.1063/1.1522750

Frenkel, D., & Smit, B. (2002). *Understanding Molecular Simulation: From Algorithms to Applications* (2nd ed.). Orlando: Academic Press.

Groot, R., & Warren, P. (1997). Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. *The Journal of Chemical Physics, 107*(11), 4423-4435. doi: citeulike-article-id:2316331

Hoogerbrugge, P. J., & Koelman, J. M. V. A. (1992). Simulating Microscopic Hydrodynamic Phenomena with Dissipative Particle Dynamics. *EPL (Europhysics Letters), 19*(3), 155.

Huber, D. E., Markel, M. L., Pennathur, S., & Patel, K. D. (2009). Oligonucleotide hybridization and free-solution electrokinetic separation in a nanofluidic device. *Lab on a Chip, 9*(20), 2933-2940.

Jellema, L. C., Mey, T., Koster, S., & Verpoorte, E. (2009). Charge-based particle separation in microfluidic devices using combined hydrodynamic and electrokinetic effects. Lab on a Chip, 9(13), 1914-1925.

Kang, K., Choi, J., Nam, J. H., Lee, S. C., Kim, K. J., Lee, S.-W., & Chang, J. H. (2009). Preparation and characterization of chemically functionalized silica-coated magnetic nanoparticles as a DNA separator. *Journal of Physical Chemistry B, 113*(2), 536-543.

Larson, R. G., Perkins, T. T., Smith, D. E., & Chu, S. (1997). Hydrodynamics of a DNA molecule in a flow field. [Article]. *Physical Review E, 55*(2), 1794-1797. doi: 10.1103/PhysRevE.55.1794

Liu, M., Meakin, P., & Huang, H. (2007). Dissipative particle dynamics simulation of multiphase fluid flow in microchannels and microchannel networks. *Physics of Fluids, 19*(3), 033302.

Pan, H., Ng, T. Y., Li, H., & Moeendarbary, E. (2010). Dissipative particle dynamics simulation of entropic trapping for DNA separation. *Sensors and Actuators A: Physical, 157*(2), 328-335. doi: DOI: 10.1016/j.sna.2009.11.027

Perkins, T., Smith, D., & Chu, S. (1994). Direct observation of tube-like motion of a single polymer chain. *Science, 264*(5160), 819-822. doi: 10.1126/science.8171335

Perkins, T., Smith, D., Larson, R., & Chu, S. (1995). Stretching of a single tethered polymer in a uniform flow. *Science, 268*(5207), 83-87. doi: 10.1126/science.7701345

Pivkin, I. V., Caswell, B., & Karniadakis, G. E. (2011). *Dissipative Particle Dynamics*: John Wiley & Sons, Inc.

Pivkin, I. V., & Karniadakis, G. E. (2005). A new method to impose no-slip boundary conditions in dissipative particle dynamics. *Journal of Computational Physics, 207*(1), 114-128. doi: DOI: 10.1016/j.jcp.2005.01.006

Rapaport, D. C. (2004). *The Art of Molecular Dynamics Simulation* (2 ed.). Cambridge: Cambridge University Press.

Revenga, M., Zúñiga, I., & Español, P. (1999). Boundary conditions in dissipative particle dynamics. *Computer Physics Communications, 121-122*, 309-311. doi: 10.1016/S0010-4655(99)00341-0

Revenga, M., Zúñiga, I., Español, P., & Pagonabarraga, I. (1998). Boundary Models in DPD. *International Journal of Modern Physics C, 9*(8), 1319-1328. doi: 10.1142/S0129183198001199

Satoh, A. (2011). *Introduction to practice of molecular simulation: Molecular dynamics, Monte Carlo, Brownian dynamics, Lattice Boltzmann, dissipative particle dynamics* (1st ed.). Amsterdam ; Boston: Elsevier.

Smith, D. E., Babcock, H. P., & Chu, S. (1999). Single-Polymer Dynamics in Steady Shear Flow. [Article]. *Science, 283*(5408), 1724.

Symeonidis, V., Karniadakis, G., & Caswell, B. (2005). Dissipative Particle Dynamics Simulations of Polymer Chains: Scaling Laws and Shearing Response Compared to DNA Experiments. *Physical Review Letters, 95*(7), 076001.

Symeonidis, V., Karniadakis, G. E., & Caswell, B. (2006). Schmidt number effects in dissipative particle dynamics simulation of polymers. [Article]. *Journal of Chemical Physics, 125*(18), 184902. doi: 10.1063/1.2360274

Underhill, P. T., & Doyle, P. S. (2004). On the coarse-graining of polymers into bead-spring chains. *Journal of Non-Newtonian Fluid Mechanics, 122*(1-3), 3-31. doi: 10.1016/j.jnnfm.2003.10.006

Willemsen, S. M., Hoefsloot, H. C. J., & Iedema, P. D. (2000). No-slip boundary condition in dissipative particle dynamics. [Article]. *International Journal of Modern Physics C, 11*(5), 881-890.

APPENDICES

APPENDIX A

Matlab Functions and Usage

Matlab Usage

For the initial conditions, the initial position and initial velocity are both created as two separate functions. These two files contain positions and velocity for all fluid, wall and DNA particles. These functions are then called into the main script which contains the algorithm. The forces are written in three separate function files. Each force is called depending on which particle is being computed. When the fluid particles, i =1 to N, is being calculated, the force that contains the fluid-fluid, fluid-wall and fluid-polymer particle interactions are called. Similarly, when the polymer particles, i =Ndtot+1 to Ntot, is being calculated, the force that contains the polymer-fluid, polymer-wall and polymer-polymer particle interactions are called along with the force function that contains the polymer spring forces. Finally, the script file is created which contains the main algorithm which is the modified velocity-Verlet. The modified boundary condition with no-slip and modified periodic boundary condition is coded in the script as well.

The script and the function files are saved as *.m files. The command window and workspace can be saved as *.mat file where the values calculated through the simulation can be retrieved. These values can also be transferred for the *.mat file to Excel if needed.

Each vector that is calculated as a matrix is computed in both x and y direction. For example, the velocity vectors v(1,i) and v(2,1) are in the x- and y- directions respectively.

Matlab Functions

Matlab has a lot of functions in-built into the software, but these are some of the few ones that are used in the code. The function *global* makes the variables public which can be used through each function called. Also different copies of the script can be created to do multiple simulations at the same with different parameters and same force and initial condition *.m function files. The second function used is called *rand(n)* which generated a uniformly distributed random numbers in a n x n matrix. The random numbers are between the open interval (0,1). This function is used to generate random numbers for the Maxwell-Boltzmann velocity distribution for the boundary conditions, initial velocity and θ_{ij}, parameter for the random force.

APPENDIX B

Relation of parameter s and DNA migration

Table 4 Relation between Weighting Function and Migration Distance.

Beads	5.00				10.00				20.00			
s	0.50	1.00	1.50	2.00	0.50	1.00	1.50	2.00	0.50	1.00	1.50	2.00
Time, t												
0	-9.99	-9.99	-9.99	-9.99	-9.99	-9.99	-9.99	-9.99	-9.99	-9.99	-9.99	-9.99
25	-8.80	-7.87	-11.08	-8.55	-8.12	-10.26	-9.83	-9.15	-10.28	-11.22	-9.22	-10.65
50	-11.65	-8.52	-9.44	-8.37	-8.07	-10.22	-8.42	-8.88	-10.90	-11.07	-9.09	-8.40
75	-11.72	-10.23	-8.82	-10.73	-8.34	-7.74	-7.90	-7.64	-11.18	-8.29	-11.80	-6.24
100	-11.66	-11.90	-9.90	-11.03	-6.95	-8.30	-6.62	-9.12	-9.85	-7.65	-12.02	-8.39
125	-9.10	-12.91	-10.07	-9.47	-8.53	-6.18	-5.57	-8.91	-7.42	-6.96	-13.97	-9.46
150	-9.24	-12.99	-11.32	-10.02	-9.08	-5.32	-4.62	-8.46	-7.11	-9.17	-11.52	-11.21
175	-10.68	-13.31	-10.38	-11.18	-7.90	-5.75	-4.71	-8.79	-8.57	-7.63	-11.87	-11.74
200	-10.83	-13.28	-10.13	-10.09	-7.16	-5.07	-4.61	-8.12	-9.58	-8.06	-11.76	-10.47
225	-12.51	-11.37	-8.92	-14.04	-7.67	-7.14	-5.14	-9.17	-9.61	-6.14	-10.19	-10.09
250	-12.47	-7.39	-9.05	-13.36	-7.83	-8.36	-4.86	-5.59	-12.39	-4.79	-9.26	-10.05
275	-13.36	-8.64	-8.08	-11.21	-7.69	-8.55	-3.35	-5.44	-11.35	-1.59	-10.34	-8.13
300	-13.76	-8.68	-8.44	-10.39	-9.17	-8.53	-4.72	-6.64	-11.79	-0.94	-11.51	-10.02
325	-13.21	-7.26	-8.84	-8.48	-9.17	-8.45	-5.38	-4.78	-10.95	-1.97	-11.37	-10.30
350	-13.60	-6.97	-8.95	-10.60	-9.82	-8.48	-6.07	-6.45	-10.03	-3.78	-11.60	-11.10
375	-12.59	-6.18	-10.81	-8.84	-7.59	-8.87	-7.79	-6.30	-10.64	-2.77	-11.59	-12.65
400	-12.68	-5.84	-11.33	-8.62	-6.54	-10.25	-10.27	-6.18	-11.67	-2.81	-10.43	-12.22
425	-10.92	-4.82	-12.51	-9.85	-6.18	-11.42	-10.58	-6.77	-11.74	-3.65	-11.63	-14.11
450	-11.47	-4.11	-12.11	-11.27	-6.51	-12.60	-10.46	-7.79	-10.37	-4.36	-9.01	-13.39
475	-12.61	-3.32	-11.13	-11.44	-7.33	-13.64	-10.27	-7.49	-9.93	-4.43	-9.23	-11.56
500	-12.14	-4.21	-10.84	-10.70	-7.05	-13.58	-11.23	-9.11	-8.67	-5.69	-9.18	-9.10

Viscosity and Schmidt Number

Table 5 Properties of DPD System (Fan et al., 2006)

Properties	Conventional ($s=2$)	Modified ($s=1/2$)
Diffusivity, D	$\frac{45K_BT}{2\pi\gamma\rho r_c^3}$	$\frac{315K_BT}{64\pi\rho r_c^3}$
Viscosity, η	$\frac{\rho D}{2}+\frac{2\pi\gamma\rho^2 r_c^5}{1575}$	$\frac{\rho D}{2}+\frac{512\pi\gamma\rho^2 r_c^5}{51975}$
Schmidt number, Sc	$\frac{1}{2}+\frac{(2\pi\gamma\rho r_c^4)^2}{70875K_BT}$	$\frac{1}{2}+\frac{(2\pi\gamma\rho r_c^4)^2}{1999K_BT}$

APPENDIX C

Matlab Code

```
%%%%%%%%%%%%%%%%%%%
%   Main Algorithm script
%%%%%%%%%%%%%%%%%%%


tic
clc
clear all

global kBT aff aww afw rc rc2 rcw s delt
global LX LY LX1 LX2 LY1 LY2
global Nwall Nwall2 Q Qwall Ndtot Nptot Ntot
global velmxd
global A Qm1 nx ny nwallx nwally
global sigma gamma rho md g lambda
global afp apw Np Npb lseg nyp leff L lp kBTp


% PARAMETERS
%DPD constants

dm=2;                   %Dimensions
kBT=1;                  %=kB*Temp
sigma=3;
gamma=4.5;
lambda=0.65;
rho=4;
aff=(75*kBT)/rho;       % =18.75
aww=5.0;                % =5
afw=sqrt(aff*aww);      % =9.682
rc=1;
rc2=rc^2;
s=2;
rl=1.5;                 % Verlet Neighbour List Method r<rc<=rl
delt=0.02;              %time step
tf=300;                  %Number of time steps (t>1350)
ti=0.;                  %inital time
g=0.05;                    %Driving force in x direction
M=1;                    % Mass density of DPD particles
md=1;
velmxd=sqrt(2*kBT/md);  % Maximum velocity of particles


%-------------------------------------------------------------
% Initial Conditions
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DPD Fluid Particles
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

N=1458;                      % Number of particles
LX=12;
LY=30;
rcw=0.005*LY;
ndensdn=1;                   % Non-dimensional Number density of DPD
dc=0.4;                      % Diameter of Particle
ndensd=N/(LX*LY);            % Number density of dpd particles
vdensd=ndensdn*pi/4;         % Volumetric Fraction
Q=2*sqrt(N/2);
A=sqrt(1/ndensd);           % Number density
Qm1=(Q-1);
nx=(LX)/Q;
ny=(LY)/Q;
LX1=LX/2;
LX2=-(LX/2);
LY1=LY/2;
LY2=(-LY/2);
bc1=(LY1)-rcw;
bc2=(LY2)+rcw;
bins=5;
tbins=bins+2;               %Total Bins
nybins=(LY-(2*rcw))/bins;
nymat=[(LY2+rcw/2) (LY2+rcw+(nybins/2):nybins:LY1-rcw) (LY1-rcw/2)];
bnbtm=rcw-(LY1);    %Bottom bin near wall ny=0.005*LY
bntp=(LY1)-rcw;     %Top bin near wall ny=0.005*LY


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Wall Setup
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nwall=400;                  %Number of Wall particles
Nwall2=Nwall/2;
Qwall=sqrt(Nwall);
nwallx=LX/Nwall2;
nwally=LY/Nwall2;
Ndtot=Nwall+N;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DNA Particles
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Np=1;                        % Number of strands
Npb=10;                      %Number of beads in the strand=81
leff=0.053;                  %effective length;
kBTp=1;                      %uJ kBTp=4.115*10^(-14)erg Erg=1*10^-7 J
L=67.2;                      %total length of the DNA strand 67.2 um
lp=0.81;                     % for Npb=81 and lp=L/(Npb-1);
lseg=0.4;                    % Initial distance between beads
mp=0.25;                     % g/cm3
```

```
viscp=2.588;
nvp=1.235;                  %cP um/s
Nptot=Np*Npb;               % Total number of beads in all strands
Ntot=Ndtot+Nptot;
afp=2;                      % repulsion force between fluid and polymer
apw=sqrt(2*aww);
nyp=LY/Np;

%Initiate Arrays
newv=zeros(2,Ntot);
newF=zeros(2,Ntot);   predv=zeros(2,Ntot);
vavgd2=zeros(1,tbins); vavgd=zeros(1,tbins);
vnum=zeros(1,tbins);
diffrpl=zeros(2,Nptot);
absrpl=zeros(1,Nptot);
constl=zeros(1,Nptot);
vecl=zeros(2,Nptot);

%Initiate Conditions
r=initposd(N,Nwall);
newr=initposd(N,Nwall);
v=initveld(N,Nwall,Np,Npb);
vnum(1,1:tbins)=0;
vavgd(1,1:tbins)=0;
F=0; Fp=0; Fps=0;

%Initial Force
for i=1:1:N
    F=F+force(r,v,i,N,Nwall);
end

for i=Ndtot+1:1:Ndtot+Nptot
    Fp=Fp+forcefp(r,v,i,N,Nwall);
    Fps=Fps+forcepp(r,i,N,Nwall);

end
F=F+Fp+Fps;


%Modified Velocity Verlet

for t=ti:delt:tf
    t
    for i=1:1:N
        for k=1:2
            if abs(r(2,i))<bc1
            newr(k,i)=r(k,i)+delt*v(k,i)+(1/2)*delt^2*(1/M)*F(k,i);
            predv(k,i)=v(k,i)+ lambda*delt*(1/M)*F(k,i);
            else
            newr(k,i)=r(k,i)+delt*v(k,i);
            predv(k,i)=v(k,i);
            end
        end

        %Setting Periodic Boundary Conditions
```

```
        if newr(1,i)>= LX1
            newr(1,i)=newr(1,i)-LX;
        elseif newr(1,i)<= LX2
            newr(1,i)=newr(1,i)+LX;
        end

    end

% When particles are close to the wall particles

    for i=1:1:N

    if newr(2,i)>=bc1        % Top wall
        n=-1;
        vRx=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
        vRy=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
        if abs(v(1,i))>=abs(vRx)
            newv(1,i)=vRx;
        else
            newv(1,i)=v(1,i);
        end
        if abs(v(2,i))>=abs(vRy)
            newv(2,i)=vRy+n*(sqrt((n*vRy)^2)-(n*vRy));
            if newv(2,i)>0
                disp('positive at upper wall')
                newv(2,i)=newv(2,i)*(-1);
            end
        else
            newv(2,i)=n*abs(v(2,i));
        end

        newF=zeros(2,Ntot);
        newr(2,i)=bc1;

    elseif newr(2,i)<=bc2     % Bottom wall
        n=1;
        vRx=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
        vRy=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
        if abs(v(1,i))>=abs(vRx)
            newv(1,i)=vRx;
        else
            newv(1,i)=v(1,i);
        end
        if abs(v(2,i))>=abs(vRy)
            newv(2,i)=vRy+n*(sqrt((n*vRy)^2)-(n*vRy));
            if newv(2,i)<0
                disp('negative at lower wall')
                newv(2,i)=newv(2,i)*(-1);
            end
        else
            newv(2,i)=n*abs(v(2,i));
        end
        newF=zeros(2,Ntot);
        newr(2,i)=bc2;
```

```
    else
        newF=force(newr,predv,i,N,Nwall);
        newv(1,i)=v(1,i)+(1/2)*delt*(1/M)*(F(1,i)+newF(1,i));
        newv(2,i)=v(2,i)+(1/2)*delt*(1/M)*(F(2,i)+newF(2,i));
    end

% Velocity check to ensure velocity does not exceed
%  maximum velocity velmxd

        vavg=(newv(1,i))^2+(newv(2,i))^2;
        if vavg > velmxd^2
            vavg2=sqrt(velmxd^2/vavg);
            newv(1,i)=newv(1,i)*vavg2;
            newv(2,i)=newv(2,i)*vavg2;
        end

        r(1,i)=newr(1,i);
        r(2,i)=newr(2,i);
        v(1,i)=newv(1,i);
        v(2,i)=newv(2,i);
        F(1,i)=newF(1,i);
        F(2,i)=newF(2,i);

    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DNA Particle interaction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i=Ndtot+1:1:Ndtot+Nptot
        for k=1:2

            if abs(r(2,i))<bc1
            newr(k,i)=r(k,i)+delt*v(k,i)+(1/2)*delt^2*(1/M)*F(k,i);
            predv(k,i)=v(k,i)+ lambda*delt*(1/M)*F(k,i);
            else
            newr(k,i)=r(k,i)+delt*v(k,i);
            predv(k,i)=v(k,i);
            end
        end

% Left beads are recorded and the right beads are adjusted with
%   length less than or equal to 0.01

diffrpl(1,i)=newr(1,i)-newr(1,i-1);
    if diffrpl(1,i) > LX1
        diffrpl(1,i)=diffrpl(1,i)-LX;
    elseif diffrpl(1,i) < LX2
        diffrpl(1,i)=LX - abs(diffrpl(1,i));
    end

diffrpl(2,i)=newr(2,i)-newr(2,i-1);
absrpl(i)=sqrt((diffrpl(1,i)).^2+(diffrpl(2,i)).^2);
constl(i)=absrpl(i)/lp;
vecl(1,i)=diffrpl(1,i)./absrpl(i);
```

```
vecl(2,i)=diffrpl(2,i)./absrpl(i);

% Constraint to prevent length of segments near fixed end exceeding lp
if i~=Ndtot+1
    if abs(constl(i)) > 0.95
        newr(1,i)=newr(1,i-1)+lp*vecl(1,i);
        newr(2,i)=newr(2,i-1)+lp*vecl(2,i);
    end
end

        %Setting Periodic Boundary Conditions
        if newr(1,i)>= LX1
            newr(1,i)=newr(1,i)-LX;
        elseif newr(1,i)<= LX2
            newr(1,i)=newr(1,i)+LX;
        end

    end



    for i=Ndtot+1:1:Ndtot+Nptot

    if newr(2,i)>=bc1                % Top wall
        n=-1;
        vRx=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
        vRy=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
        newv(1,i)=vRx;
        newv(2,i)=vRy+n*(sqrt((n*vRy)^2)-(n*vRy));
            if newv(2,i)>0
                disp('positive at upper wall')
                newv(2,i)=newv(2,i)*(-1);
            end

        newF=zeros(2,Ntot);   %function Force
        newr(2,i)=0;

    elseif newr(2,i)<=bc2          % Bottom wall
        n=1;
        vRx=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
        vRy=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
        newv(1,i)=vRx;
        newv(2,i)=vRy+n*(sqrt((n*vRy)^2)-(n*vRy));
        if newv(2,i)<0
            disp('negative at lower wall')
            newv(2,i)=newv(2,i)*(-1);
        end

        newF=zeros(2,Ntot);   %function Force
        newr(2,i)=0;

    else
        newF=forcefp(newr,predv,i,N,Nwall)+forcepp(newr,i,N,Nwall);
        newv(1,i)=v(1,i)+(1/2)*delt*(1/M)*(F(1,i)+newF(1,i));
        newv(2,i)=v(2,i)+(1/2)*delt*(1/M)*(F(2,i)+newF(2,i));
    end
```

```
        vavg=(newv(1,i))^2+(newv(2,i))^2;
        if vavg > velmxd^2
            vavg2=sqrt(velmxd^2/vavg);
            newv(1,i)=newv(1,i)*vavg2;
            newv(2,i)=newv(2,i)*vavg2;
        end

        r(1,i)=newr(1,i);
        r(2,i)=newr(2,i);
        v(1,i)=newv(1,i);
        v(2,i)=newv(2,i);
        F(1,i)=newF(1,i);
        F(2,i)=newF(2,i);

    End

leastx=r(1,Ndtot+1);
for j=0:1:Npb-2
    if leastx<=r(1,Ndtot+2+j)
        leastx=leastx;
    else
        leastx=r(1,Ndtot+2+j);
    end
end

mostx=r(1,Ndtot+1);
for k=0:1:Npb-2
    if mostx>=r(1,Ndtot+2+k)
        mostx=mostx;
    else
        mostx=r(1,Ndtot+2+k);
    end
end

extx=mostx-leastx;
    if abs(extx)>LX1
         extx=mostx+leastx;
    end

        fid = fopen('ext.txt', 'a'); % Opening output file
        fprintf(fid,'%-07.4f %-07.4f\r\n',t, extx); %writing value file
        fclose(fid);    %Closing output file


ry=r(2,Ndtot+1:Ntot);
        fid = fopen('tvsrneg13.txt', 'a');
        fprintf(fid,'%-4.2f\r\n',t);
        fprintf(fid,'%-07.4f\r\n',ry);
        fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%
% PLOTS
%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Plotting the particle movement through the channels
figure(1)
plot(r(1,N+1:N+Nwall),r(2,N+1:N+Nwall),'o','LineWidth',0.2, ...
    'MarkerSize',5,'MarkerEdgeColor','k','MarkerFaceColor','b')
hold on
% whitebg('white')
%     set(gcf,'Color',[0.5,1,0.6])
plot(r(1,1:N),r(2,1:N),'o','MarkerSize',6, ...
     'MarkerEdgeColor','k','MarkerFaceColor','y')
% plot(r(1,200),r(2,200),'>','MarkerSize',6, ...
%     'MarkerEdgeColor','k','MarkerFaceColor','r')
plot(r(1,Ndtot+1:Ntot),r(2,Ndtot+1:Ntot),'-ok','MarkerSize',5, ...
     'MarkerEdgeColor','k','MarkerFaceColor','r')
text(r(1,Ndtot+1),r(2,Ndtot+1),num2str(Ndtot+1))
text(r(1,Ndtot+2),r(2,Ndtot+2),num2str(Ndtot+2))
text(r(1,Ndtot+3),r(2,Ndtot+3),num2str(Ndtot+3))
text(r(1,Ndtot+4),r(2,Ndtot+4),num2str(Ndtot+4))
text(r(1,Ndtot+5),r(2,Ndtot+5),num2str(Ndtot+5))
% axis tight
axis([-LX/2 LX/2 -LY/2 LY/2])
drawnow
hold off


% Plotting the averaged velocity in each bin over a set time step
for i=1:1:N
        if r(2,i)<=bnbtm   %&& r(2,i)>=(LY2)
            vnum(1)=vnum(1)+1;
            vavgd(1)=vavgd(1)+v(1,i);
        elseif r(2,i)>=bntp %&& r(2,i)<=(LY1)
            vnum(tbins)=vnum(tbins)+1;
            vavgd(tbins)=vavgd(tbins)+v(1,i);
        end
end
for p=1:1:bins
sect=((p)*nybins)-(LY1)+rcw;
sect2=((p-1)*nybins)-(LY1)+rcw;
    for i=1:1:N
        if r(2,i)<=sect && r(2,i)>sect2
            vnum(p+1)=vnum(p+1)+1;
            vavgd(p+1)=vavgd(p+1)+v(1,i);
        end
    end

end


for tm=1:1:10
    if t==(tf/10)*tm
        vavgd2(:)=vavgd(:)./vnum(:);
        vavgd2(isnan(vavgd2))=0;
        vtot=sum(vavgd2,2);
        vnorm=vtot/tbins;
        vovnrm(1,:)=vavgd2(:)/vnorm;

figure(4);
```

```
plot(nymat(:),vovnrm(1,:),'o','MarkerSize',6, ...
     'MarkerEdgeColor','k','MarkerFaceColor','k')
xlabel('BINS')
ylabel('VELOCITY /AVERAGE VELOCITY')
axis tight
% axis([-15 15 -1 3])
set(gca,'XMinorTick','on','YMinorTick','on')
drawnow


vnormtxt=[nymat; vovnrm];
% open the file with write permission
fid = fopen('posnorm1.txt', 'a'); % Opening output file
fprintf(fid,' \r\n');
fprintf(fid,'%-4.2f\r\n',t);
fprintf(fid,'%-07.4f %-07.4f\r\n',vnormtxt);  %writing to output file
fclose(fid);    %Closing output file

    else
        continue
    end
end

      if t<=10
              vnum(1,1:tbins)=0;
              vavgd(1,1:tbins)=0;
      end

end


figure(3);
plot(nymat(:),vavgd2(:),':k','LineWidth',2)
xlabel('BINS')
ylabel('AVERAGE VELOCITY')
axis tight
% axis([-1.5 1.5 -1.5 1.5])
set(gca,'XMinorTick','on','YMinorTick','on')

toc


%%%%%%%%%%%%%%%%%%
%  Initial Positions
%%%%%%%%%%%%%%%%%%

function [ r ] = initposd( N,Nwall )

global LX LY Np Npb Ntot
global Q Qm1 nx ny nwallx Nwall2
global LX2 LY2 lseg nyp

%Initiate Arrays
ri=zeros(2,Q);  r=zeros(2,Ntot);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Fluid Particle Setup
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

i=0;

for s=0:2:Qm1-1
    ri(1,s+1)=(s*nx)-(LX/2)+0.05;

    if ri(1,s+1)>(LX/2)
       break
    end

    for p=0:2:Qm1-1
        ri(2,p+1)=(p*ny)-(LY/2)+0.05;

        if ri(2,p+1)>(LY/2)
           break
        end

            i=i+1;
            r(1,i)=ri(1,s+1);       % X-Position of the particles
            r(2,i)=ri(2,p+1);       % Y-Position of the particles
    end
end


for s=1:2:Qm1
    ri(1,s+1)=(s*nx)-(LX/2)+0.05+(nx/20);

    if ri(1,s+1)>(LX/2)
       break
    end

    for p=1:2:Qm1
        ri(2,p+1)=(p*ny)-(LY/2)+0.05+(ny/8);

        if ri(2,p+1)>(LY/2)
           break
        end

        i=i+1;
        r(1,i)=ri(1,s+1);       % X-Position of the particles
        r(2,i)=ri(2,p+1);       % Y-Position of the particles
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Wall Setup
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for i=N+1:1:(N+Nwall2)
    r(1,i)= (i*nwallx)-(LX/2)+0.001-((N+1)*nwallx);
        if r(1,i)>(LX/2)
            break
        end
    r(2,i)=-(LY/2);
        if r(2,i)>(LY/2)
            break
        end
end

for i=N+1:1:(N+Nwall2)
    r(1,Nwall2+i)= (i*nwallx)-(LX/2)+0.001-((N+1)*nwallx);
        if r(1,Nwall2+i)>(LX/2)
            break
        end
    r(2,Nwall2+i)=(LY/2);
        if r(2,Nwall2+i)>(LY/2)
            break
        end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DNA Setup
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

i=N+Nwall;
rpi(1,1)=LX2;

for p=1:1:Np
    rpi(2,p+1)=LY2+((p-1)*nyp)+(nyp/2);

    for s=1:1:Npb
    rpi(1,s+1)=rpi(1,s)+lseg;

    if rpi(1,s+1)>(LX/2)
       break
    end

        i=i+1;
        r(1,i)=rpi(1,s+1);      % X-Position of the particles
        r(2,i)=rpi(2,p+1);      % Y-Position of the particles
    end
end

end


%%%%%%%%%%%%%%%%%%%
%  Initial Velocity
%%%%%%%%%%%%%%%%%%%

function [ v ] = initveld( N,Nwall,Np,Npb )
```

```
global kBT md Nptot Ndtot Ntot
global velmxd

%Initiate Arrays
v=zeros(2,Ntot);

for i=1:1:N
    v(1,i)=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
    v(2,i)=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);

% Velocity check to ensure velocity does not exceed max velocity velmxd
vavg=(v(1,i))^2+(v(2,i))^2;
    if vavg > velmxd
        vavg=sqrt(velmxd/vavg);
        v(1,i)=v(1,i)*vavg;
        v(2,i)=v(2,i)*vavg;
    end
end

% To set Total Momentum equals Zero
momxd=0;
momyd=0;

for i=1:1:N
    momxd=momxd+v(1,i);
    momyd=momyd+v(2,i);
end

momxd=momxd/N;
momyd=momyd/N;

for i=1:1:N
    v(1,i)=v(1,i)-momxd;
    v(2,i)=v(2,i)-momyd;
end


%%%%%%%%%%%%%%%%%%%
% Wall Velocity
%%%%%%%%%%%%%%%%%%%
for i=N+1:1:Ndtot

    v(1,i)=0;
    v(2,i)=0;
end

%%%%%%%%%%%%%%%%%%%
% DNA Velocity
%%%%%%%%%%%%%%%%%%%
for i=Ndtot+1:1:Ndtot+Nptot

    v(1,i)=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
    v(2,i)=sqrt((-2)*(kBT/md)*log(rand))*cos(2*pi*rand);
```

```
% Velocity check to ensure velocity does not exceed max velocity velmxd
vavg=(v(1,i))^2+(v(2,i))^2;
    if vavg > velmxd
        vavg=sqrt(velmxd/vavg);
        v(1,i)=v(1,i)*vavg;
        v(2,i)=v(2,i)*vavg;
    end
end


%%%%%%%%%%%%%%%%%%
%   DPD Fluid Forces
%%%%%%%%%%%%%%%%%%

function [ F ] = force( r,v,i,N,Nwall )

global kBT aff afw afp rc rc2 s delt
global LX LX1 LX2
global Ndtot Nptot Ntot
global sigma gamma g

%Initiate Arrays
diffr=zeros(2,N); diffv=zeros(2,N);
absr=zeros(1,N); absr2=zeros(1,N); absv=zeros(1,N);
diffrvec=zeros(2,N); diffvvec=zeros(2,N);
FCon=zeros(2,N); FDis=zeros(2,N);
FRan=zeros(2,N);
F=zeros(2,Ntot);
dotrv=zeros(2,N);
Fint=zeros(2,N); Fintw=zeros(2,N); Fintp=zeros(2,N);
Fext(1,1:N)=g;
Fext(2,1:N)=0;

Fint(1,i)=0;
Fint(2,i)=0;
Fintw(1,i)=0;
Fintw(2,i)=0;
Fintp(1,i)=0;
Fintp(2,i)=0;

for j=1:1:N

    if j==i
        continue
    end

    %Distance between two particles with x and v components
    diffr(1,i)=r(1,i)-r(1,j);
      if diffr(1,i) > LX1       % Periodic Boundary Conditions
            diffr(1,i)=diffr(1,i)-LX;
      elseif diffr(1,i) < LX2
            diffr(1,i)=LX - abs(diffr(1,i));
      end
```

```
    diffr(2,i)=r(2,i)-r(2,j);
      if abs(diffr(1,i))>rc
            continue
      end
      if abs(diffr(2,i))>rc
            continue
      end
    absr(i)=sqrt((diffr(1,i)).^2+(diffr(2,i)).^2);
    absr2(i)=(absr(i)).^2;
      if absr2(i)>rc2
            continue
      end

    diffrvec(1,i)=diffr(1,i)./absr(i);
    diffrvec(2,i)=diffr(2,i)./absr(i);

    %Velocity between two particles with x and v components
    diffv(1,i)=v(1,i)-v(1,j);
    diffv(2,i)=v(2,i)-v(2,j);
    absv(i)=sqrt((diffv(1,i)).^2+(diffv(2,i)).^2);
    diffvvec(1,i)=diffv(1,i)./absv(i);
    diffvvec(2,i)=diffv(2,i)./absv(i);


    %Conservative Force- Repulsive Force
    FCon(1,i)=aff*(1-absr(i)).*diffrvec(1,i);
    FCon(2,i)=aff*(1-absr(i)).*diffrvec(2,i);

    if abs(absr(i))<=rc
        wD=(1-absr(i)/rc)^s;
    else
        wD=0;
    end

    wR=sqrt(wD);
    theta= sqrt((-2)*log(rand))*cos(2*pi*rand);

        if theta > 6
            theta=sign(theta)*6;
        end

    %Dissipative Force
    dotrv(i)=(diffrvec(1,i).*diffv(1,i))+(diffrvec(2,i).*diffv(2,i));
    FDis(1,i)=-gamma*wD*dotrv(i).*diffrvec(1,i);
    FDis(2,i)=-gamma*wD*dotrv(i).*diffrvec(2,i);

    %Random Force
    FRan(1,i)=sigma*wR*theta*diffrvec(1,i);
    FRan(2,i)=sigma*wR*theta*diffrvec(2,i);

    %Internal Forces
    Fint(1,i)=Fint(1,i)+(FCon(1,i)+FDis(1,i)+FRan(1,i));
    Fint(2,i)=Fint(2,i)+(FCon(2,i)+FDis(2,i)+FRan(2,i));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%
%  Wall Particles
%%%%%%%%%%%%%%%%%%%%%%%%

for j=N+1:1:N+Nwall

    %Distance between two particles with x and v components
    diffr(1,i)=r(1,i)-r(1,j);
    if diffr(1,i) > LX1     % Periodic Boundary Conditions
        diffr(1,i)=diffr(1,i)-LX;
    elseif diffr(1,i) < LX2
        diffr(1,i)=LX - abs(diffr(1,i));
    end

    diffr(2,i)=r(2,i)-r(2,j);
    if abs(diffr(1,i))>rc
        continue

    end

    if abs(diffr(2,i))>rc
        continue
    end

    absr(i)=sqrt((diffr(1,i)).^2+(diffr(2,i)).^2);
    absr2(i)=(absr(i)).^2;
    if absr2(i)>rc2
        continue
    end

    diffrvec(1,i)=diffr(1,i)./absr(i);
    diffrvec(2,i)=diffr(2,i)./absr(i);

    %Velocity between two particles with x and v components
    diffv(1,i)=v(1,i)-v(1,j);
    diffv(2,i)=v(2,i)-v(2,j);
    absv(i)=sqrt((diffv(1,i)).^2+(diffv(2,i)).^2);
    diffvvec(1,i)=diffv(1,i)./absv(i);
    diffvvec(2,i)=diffv(2,i)./absv(i);

    %Conservative Force- Repulsive Force
    FCon(1,i)=afw*(1-absr(i)).*diffrvec(1,i);
    FCon(2,i)=afw*(1-absr(i)).*diffrvec(2,i);

    if abs(absr(i))<=rc
        wD=(1-absr(i)/rc)^s;
    else
        wD=0;
    end

    %Weight Functions and Coefficients of FD and FR
    wR=sqrt(wD);
    theta= sqrt((-2)*log(rand))*cos(2*pi*rand);
```

```
        if theta > 6
            theta=sign(theta)*6;
        end

    %Dissipative Force
    dotrv(i)=(diffrvec(1,i).*diffv(1,i))+(diffrvec(2,i).*diffv(2,i));
    FDis(1,i)=-gamma*wD*dotrv(i).*diffrvec(1,i);
    FDis(2,i)=-gamma*wD*dotrv(i).*diffrvec(2,i);

    %Random Force
    FRan(1,i)=sigma*wR*theta*diffrvec(1,i);
    FRan(2,i)=sigma*wR*theta*diffrvec(2,i);

    %Internal Forces
    Fintw(1,i)=Fintw(1,i)+(FCon(1,i)+FDis(1,i)+FRan(1,i));
    Fintw(2,i)=Fintw(2,i)+(FCon(2,i)+FDis(2,i)+FRan(2,i));

end

%%%%%%%%%%%%%%%%%%%%%
% DNA Particles
%%%%%%%%%%%%%%%%%%%%%

for j=Ndtot+1:1:Ndtot+Nptot

      diffr(1,i)=r(1,i)-r(1,j);

      if diffr(1,i) > LX1          % Periodic Boundary Conditions
            diffr(1,i)=diffr(1,i)-LX;
      elseif diffr(1,i) < LX2
            diffr(1,i)=LX - abs(diffr(1,i));
      end
    diffr(2,i)=r(2,i)-r(2,j);
      if abs(diffr(1,i))>rc     % Setting neighboring particles
            continue
      end
      if abs(diffr(2,i))>rc
            continue
      end
    absr(i)=sqrt((diffr(1,i)).^2+(diffr(2,i)).^2);
    absr2(i)=(absr(i)).^2;
      if absr2(i)>rc2
            continue
      end
    diffrvec(1,i)=diffr(1,i)./absr(i);
    diffrvec(2,i)=diffr(2,i)./absr(i);

    %Velocity between two particles with x and v components
    diffv(1,i)=v(1,i)-v(1,j);
    diffv(2,i)=v(2,i)-v(2,j);
    absv(i)=sqrt((diffv(1,i)).^2+(diffv(2,i)).^2);
    diffvvec(1,i)=diffv(1,i)./absv(i);
    diffvvec(2,i)=diffv(2,i)./absv(i);
```

```
    %Conservative Force- Repulsive Force
    FCon(1,i)=afp*(1-absr(i)).*diffrvec(1,i);
    FCon(2,i)=afp*(1-absr(i)).*diffrvec(2,i);

    if abs(absr(i))<=rc
        wD=(1-absr(i)/rc)^s;
    else
        wD=0;
    end

    wR=sqrt(wD);
    gamma=(sigma^2)/(2*kBT);
    theta= sqrt((-2)*log(rand))*cos(2*pi*rand);

        if theta > 6
            theta=sign(theta)*6;
        end

    %Dissipative Force
    dotrv(i)=(diffrvec(1,i).*diffv(1,i))+(diffrvec(2,i).*diffv(2,i));
    FDis(1,i)=-gamma*wD*dotrv(i).*diffrvec(1,i);
    FDis(2,i)=-gamma*wD*dotrv(i).*diffrvec(2,i);

    %Random Force
    FRan(1,i)=sigma*wR*theta*diffrvec(1,i);
    FRan(2,i)=sigma*wR*theta*diffrvec(2,i);

    %Internal Polymer Forces
    Fintp(1,i)=Fintp(1,i)+FCon(1,i)+FDis(1,i)+FRan(1,i)*delt^(-0.5);
    Fintp(2,i)=Fintp(2,i)+FCon(2,i)+FDis(2,i)+FRan(2,i)*delt^(-0.5);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Total Forces on Particles
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
F(1,i)=Fint(1,i)+Fext(1,i)+Fintw(1,i)+Fintp(1,i);
F(2,i)=Fint(2,i)+Fext(2,i)+Fintw(2,i)+Fintp(2,i);

end


%%%%%%%%%%%%%%%%%%
%   DPD DNA or Polymer Forces
%%%%%%%%%%%%%%%%%%


function [Fp ] = forcefp(r,v,i,N,Nwall)

global LX LX1 LX2
global kBT rc rc2 s
global Ndtot Nptot Ntot
global sigma gamma g delt
global afp apw
```

```
%Initiate Arrays
diffr=zeros(2,Ntot); diffv=zeros(2,Ntot);
absr=zeros(1,Ntot); absr2=zeros(1,Ntot); absv=zeros(1,Ntot);
diffrvec=zeros(2,Ntot); diffvvec=zeros(2,Ntot);
FCon=zeros(2,Ntot); FDis=zeros(2,Ntot);
FRan=zeros(2,Ntot);
Fp=zeros(2,Ntot);
dotrv=zeros(2,Ntot);

Fint(1,i)=0;
Fint(2,i)=0;
Fintw(1,i)=0;
Fintw(2,i)=0;
Fintp(1,i)=0;
Fintp(2,i)=0;

%%%%%%%%%%%%%%%%%%%%
% Fluid Particles
%%%%%%%%%%%%%%%%%%%%

for j=1:1:N

    %Distance between two particles with x and v components
    diffr(1,i)=r(1,i)-r(1,j);
    if diffr(1,i) > LX1
        diffr(1,i)=diffr(1,i)-LX;
    elseif diffr(1,i) < LX2
        diffr(1,i)=LX - abs(diffr(1,i));
    end

    diffr(2,i)=r(2,i)-r(2,j);
    if abs(diffr(1,i))>rc
        continue
    end

    if abs(diffr(2,i))>rc
        continue
    end

    absr(i)=sqrt((diffr(1,i)).^2+(diffr(2,i)).^2);
    absr2(i)=(absr(i)).^2;
    if absr2(i)>rc2
        continue
    end

    diffrvec(1,i)=diffr(1,i)./absr(i);
    diffrvec(2,i)=diffr(2,i)./absr(i);

    %Velocity between two particles with x and v components
    diffv(1,i)=v(1,i)-v(1,j);
    diffv(2,i)=v(2,i)-v(2,j);
    absv(i)=sqrt((diffv(1,i)).^2+(diffv(2,i)).^2);
    diffvvec(1,i)=diffv(1,i)./absv(i);
    diffvvec(2,i)=diffv(2,i)./absv(i);
```

```
    FCon(1,i)=afp*(1-absr(i)).*diffrvec(1,i);
    FCon(2,i)=afp*(1-absr(i)).*diffrvec(2,i);

    if abs(absr(i))<=rc
        wD=(1-absr(i)/rc)^s;
    else
        wD=0;
    end

    wR=sqrt(wD);
    theta= sqrt((-2)*log(rand))*cos(2*pi*rand);
         if theta > 6
            theta=sign(theta)*6;
        end

    dotrv(i)=(diffrvec(1,i).*diffv(1,i))+(diffrvec(2,i).*diffv(2,i));
    FDis(1,i)=-gamma*wD*dotrv(i).*diffrvec(1,i);
    FDis(2,i)=-gamma*wD*dotrv(i).*diffrvec(2,i);

    FRan(1,i)=sigma*wR*theta*diffrvec(1,i);
    FRan(2,i)=sigma*wR*theta*diffrvec(2,i);
    Fint(1,i)=Fint(1,i)+(FCon(1,i)+FDis(1,i)+FRan(1,i));
    Fint(2,i)=Fint(2,i)+(FCon(2,i)+FDis(2,i)+FRan(2,i));

end

%%%%%%%%%%%%%%%%%%%%%%%%
%  Wall Particles
%%%%%%%%%%%%%%%%%%%%%%%%

for j=N+1:1:N+Nwall

    %Distance between two particles with x and v components
    diffr(1,i)=r(1,i)-r(1,j);
    if diffr(1,i) > LX1
        diffr(1,i)= diffr(1,i)-LX;
    elseif diffr(1,i) < LX2
        diffr(1,i)=LX - abs(diffr(1,i));
    end

    diffr(2,i)=r(2,i)-r(2,j);
    if abs(diffr(1,i))>rc
        continue
    end

    if abs(diffr(2,i))>rc
        continue
    end


    absr(i)=sqrt((diffr(1,i)).^2+(diffr(2,i)).^2);
    absr2(i)=(absr(i)).^2;
    if absr2(i)>rc2
```

```
        continue
    end

    diffrvec(1,i)=diffr(1,i)./absr(i);
    diffrvec(2,i)=diffr(2,i)./absr(i);


    %Velocity between two particles with x and v components
    diffv(1,i)=v(1,i)-v(1,j);
    diffv(2,i)=v(2,i)-v(2,j);
    absv(i)=sqrt((diffv(1,i)).^2+(diffv(2,i)).^2);
    diffvvec(1,i)=diffv(1,i)./absv(i);
    diffvvec(2,i)=diffv(2,i)./absv(i);

    FCon(1,i)=apw*(1-absr(i)).*diffrvec(1,i);
    FCon(2,i)=apw*(1-absr(i)).*diffrvec(2,i);
    if abs(absr(i))<=rc
        wD=(1-absr(i)/rc)^s;
    else
        wD=0;
    end

    wR=sqrt(wD);
    theta= sqrt((-2)*log(rand))*cos(2*pi*rand);
        if theta > 6
            theta=sign(theta)*6;
        end

    dotrv(i)=(diffrvec(1,i).*diffv(1,i))+(diffrvec(2,i).*diffv(2,i));
    FDis(1,i)=-gamma*wD*dotrv(i).*diffrvec(1,i);
    FDis(2,i)=-gamma*wD*dotrv(i).*diffrvec(2,i);
    FRan(1,i)=sigma*wR*theta*diffrvec(1,i);
    FRan(2,i)=sigma*wR*theta*diffrvec(2,i);
    Fintw(1,i)=Fintw(1,i)+(FCon(1,i)+FDis(1,i)+FRan(1,i));
    Fintw(2,i)=Fintw(2,i)+(FCon(2,i)+FDis(2,i)+FRan(2,i));

end

%%%%%%%%%%%%%%%%%%%%%%
% DNA Particles
%%%%%%%%%%%%%%%%%%%%%%

for j=Ndtot+1:1:Ndtot+Nptot

        if j==i
            continue
        end

    %Distance between two particles with x and v components
    diffr(1,i)=r(1,i)-r(1,j);
    if diffr(1,i) > LX1
        diffr(1,i)=diffr(1,i)-LX;
    elseif diffr(1,i) < LX2
        diffr(1,i)=LX - abs(diffr(1,i));
    end
```

```
    diffr(2,i)=r(2,i)-r(2,j);
    if abs(diffr(1,i))>rc
          continue
    end
    if abs(diffr(2,i))>rc
        continue
    end

    absr(i)=sqrt((diffr(1,i))^2+(diffr(2,i))^2);
    diffrvec(1,i)=diffr(1,i)./absr(i);
    diffrvec(2,i)=diffr(2,i)./absr(i);

    %Velocity between two particles with x and v components
    diffv(1,i)=v(1,i)-v(1,j);
    diffv(2,i)=v(2,i)-v(2,j);
    absv(i)=sqrt((diffv(1,i))^2+(diffv(2,i))^2);
    diffvvec(1,i)=diffv(1,i)./absv(i);
    diffvvec(2,i)=diffv(2,i)./absv(i);

    %Conservative Force- Repulsive Force
    if abs(i-j)>4
        app=2;
    else
        app=0;
    end

    FCon(1,i)=app*(1-absr(i))*diffrvec(1,i);
    FCon(2,i)=app*(1-absr(i))*diffrvec(2,i);

    if abs(absr(i))<=rc
        wD=(1-absr(i)/rc)^s;
    else
        wD=0;
    end

    wR=sqrt(wD);
    theta= sqrt((-2)*log(rand))*cos(2*pi*rand);
        if theta > 6
            theta=sign(theta)*6;
        end

    FDis(1,i)=-gamma*wD*dot(diffrvec(1,i),diffv(1,i))*diffrvec(1,i);
    FDis(2,i)=-gamma*wD*dot(diffrvec(2,i),diffv(2,i))*diffrvec(2,i);
    FRan(1,i)=sigma*wR*theta*diffrvec(1,i);
    FRan(2,i)=sigma*wR*theta*diffrvec(2,i);
    Fintp(1,i)=Fintp(1,i)+FCon(1,i)+FDis(1,i)+FRan(1,i)*delt^(-0.5);
    Fintp(2,i)=Fintp(2,i)+FCon(2,i)+FDis(2,i)+FRan(2,i)*delt^(-0.5);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Total Forces on Particles
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fext(1,i)=g;
```

```
Fext(2,i)=0;
Fp(1,i)=Fint(1,i)+Fext(1,i)+Fintw(1,i)+Fintp(1,i);
Fp(2,i)=Fint(2,i)+Fext(2,i)+Fintw(2,i)+Fintp(2,i);
end


%%%%%%%%%%%%%%%%%%
%  DNA or Polymer Spring Forces
%%%%%%%%%%%%%%%%%%

function [ Fps ] = forcepp( r,i,N,Nwall )

global LX LX1 LX2 Ndtot Ntot
global Nptot leff lp kBTp

%Initializing Array setup
FS=zeros(2,Ntot);
Fps=zeros(2,Ntot);
Fpint=zeros(2,Ntot);
diffrp=zeros(2,Ntot);
absrp=zeros(1,Ntot);
diffrvecp=zeros(2,Ntot);

Fpint(1,i)=0;
Fpint(2,i)=0;

for j=i-1:2:i+1

if i==Ndtot+1
   j=i+1;
elseif i==Ndtot+Nptot
    j=i-1;
end

diffrp(1,i)=r(1,i)-r(1,j);
if diffrp(1,i) > LX1
    diffrp(1,i)=LX - diffrp(1,i);
elseif diffrp(1,i) < LX2
    diffrp(1,i)=abs(diffrp(1,i))-LX;
end

diffrp(2,i)=r(2,i)-r(2,j);
absrp(i)=sqrt((diffrp(1,i))^2+(diffrp(2,i))^2);
diffrvecp(1,i)=diffrp(1,i)./absrp(i);
diffrvecp(2,i)=diffrp(2,i)./absrp(i);

%Spring Force between beads in a strand
FS(1,i)=((-kBTp)/(4*leff))*(1-(absrp(i)/lp)^(-2)+(4*absrp(i)/lp)-
1)*diffrvecp(1,i);
FS(2,i)=((-kBTp)/(4*leff))*(1-(absrp(i)/lp)^(-2)+(4*absrp(i)/lp)-
1)*diffrvecp(2,i);

Fpint(1,i)=FS(1,i)+Fpint(1,i);
Fpint(2,i)=FS(2,i)+Fpint(2,i);
```

```
if j==i+1
    break
elseif i==Ndtot+Nptot
    break
end

end

Fps(1,i)=Fpint(1,i);
Fps(2,i)=Fpint(2,i);
end
```

VITA

Saumya Susan Simon

Candidate for the Degree of

Master of Science

Thesis: SIMULATION OF THE FLOW OF A SINGLE STRANDED DNA IN A CHANNEL USING DISSIPATIVE PARTICLE DYNAMICS

Major Field: Mechanical Engineering

Biographical:

Education:

Completed the requirements for the Master of Science in Mechanical Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2011.

Completed the requirements for the Bachelor of Science in Mechanical and Aerospace Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2009.

Professional Memberships:

American Institute of Aeronautics and Astronautics

Name: Saumya Susan Simon Date of Degree: December, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: SIMULATION OF THE FLOW OF A SINGLE STRANDED DNA IN A CHANNEL USING DISSIPATIVE PARTICLE DYNAMICS

Pages in Study: 107 Candidate for the Degree of Master of Science

Major Field: Mechanical Engineering

Scope and Method of Study: Separation of DNA has significant importance in understanding the genome of an organism for genetic engineering and DNA profiling for forensics. The purpose of the study was to simulate a system containing solvent and DNA particles through a pressure-driven two-dimensional microchannel using Dissipative Particle Dynamics. This computational fluid method is simulated in Matlab using the modified velocity-Verlet algorithm. The computational method DPD was modified and the simulated channel flow was compared to the theoretical flow between two-dimensional parallel plates. The boundary conditions include both solid 'frozen' particle walls and periodic boundary conditions. The DNA particles are then inserted into the channel to understand their physical properties as they migrate through a pressure-driven channel. Their extension due to stretching and folding is studied to understand the relaxation time of the DNA strand in the channel for a set of varied conditions.

Findings and Conclusions: A no-slip boundary region was constructed to enforce the wall boundary conditions and to prevent the wall penetration by DPD fluid particles. The modified DPD weighting functions resolve the low Schmidt number and low viscosity typical of DPD and increase particle interaction between DPD fluid particles. However, this modification cannot be performed when simulating DNA particles as worm-like chain models as they do not generate accurate physical properties of DNA particles. The extensions of the DNA strands are simulated under the influence of different external forces and the relaxation time was reported.

ADVISER'S APPROVAL: Dr. Khaled Sallam