A Stochastic Simulation Method Using Constraints for the Modeling of Blood Rheology

BY
KYUNG HYO KIM

DISSERTATION

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Chemical Engineering
in the Graduate College of the
University of Illinois at Chicago, 2015

Chicago, Illinois

Defense Committee:

Lewis E. Wedgewood, Chair and Advisor
Ludwig C. Nitsche
Belinda S. Akpa
Ying Liu
Michael L. Berbaum, Institute for Health Research and Policy

This dissertation is dedicated to my parents, Young Geun Kim and Heng Shin Kim.

ACKNOWLEDGMENTS

First and foremost, I deeply thank my academic advisor Dr. Lewis E. Wedgewood for his encouragement and inspirational teachings throughout my study. I have been motivated by his mathematical abilities and insight. He has been a great mentor in many ways giving me fruitful advices.

I would also like to gratefully acknowledge my committee members, Drs. Ludwig C. Nitsche, Belinda S. Akpa, and Ying Liu. Their superb conceptual understanding helped me think problems in multifaceted direction. It was a pleasure to work with Prof. Nitsche although I regret that I have not had a chance to learn programming skills from him. I was inspired by his software development.

My deepest gratitude goes to the Institute of Health Research and Policy for an opportunity to be part of such enthusiastic group that I have a lot to learn from and their financial support. My work has benefited from the advice, feedback, and encouragement of many people, in particular, my committee member, Dr. Michael Berbaum, and Dr. Yoonsang Kim at IHRP. I appreciate them for generously sharing wisdom and giving technical supports. It was truly an enriching experience working at IHRP.

I thank Dr. Donald Hedeker, Dr. Kevin Berbaum, and the aforementioned Dr. Ludwig C. Nitsche and Dr. Michael Berbaum, all of whom I feel privileged for the opportunity to work with. Their perpetual energy and enthusiasm in research motivated me in my study. Also, thanks to coordinators in both departments I have worked, Karen and Amy.

I would like to acknowledge the following people for their magnificent support. My colleage, Manuela A. A. Ayee, always shown interest in my work and our unlimited discussions have given me inspirations. I shall always miss my dear colleages (HyeRan, Mali, Xiaoru, Nihal, Sean, Alex), undergraduates (Xin, Vijeta, Nicole), and many others including my dear friends in Korea. Special thanks also go to Mark C. Simmons, Yoonsang, Hajwa, and Manuela for supporting me to go through hardship.

ACKNOWLEDGMENTS

Last but definitely not least, I thank my parents, my brother, my sister, brother-in-law, my nephew (Andrew HeeSung), and all my extended family members for their unconditional extraordinary support. Especially, without Joon Sung, I would never have dreamed of exploring the bigger world. Thank you for your appreciation gift.

KKim

TABLE OF CONTENTS

CHAPTER **PAGE**

1. INTRODUCTION 1
   1.1 Motivation and Purpose of the Study 1
   1.2 Organization of the Dissertation 4

2. LITERATURE REVIEWS 5
   2.1 Blood Rheology 5
   2.2 Simulation Technique 8
      2.2.1 Monte Carlo (MC) 8
      2.2.2 Molecular Dynamics (MD) 9
      2.2.3 Brownian Dynamics (BD) 11

3. METHODOLOGY 14
   3.1 Kinetic Theory: Diffusion Equation 14
      3.1.1 The Equation of Motion 14
      3.1.2 The Equation of Continuity 19
   3.2 Method of Constraint with Lagrange Multiplier 21
   3.3 Model Development 25
      3.3.1 Inter-particle Forces and Constraint Setup for Each Models 28
   3.4 Stress Tensor 39
   3.5 Material Properties 42
      3.5.1 Shear Flow 42
      3.5.2 Stress Relaxation after Sudden Shearing Displacement 43
      3.5.3 Shear Free (Elongational) Flow 44
   3.6 Radius of Gyration 44

4. SIMULATION 46

5. RESULTS and DISCUSSION 49
   5.1 Three-Bead-Spring Ring Model with One Constraint 49
      5.1.1 Stress Relaxation 49
      5.1.2 Start-up Shear Flow 50
      5.1.3 Steady Shear Flow 51
      5.1.4 Capillary Flow 54
      5.1.5 Steady Shear Free Flow 57
   5.2 Three-Bead-Spring Ring Model with Two Constraints 58
      5.2.1 Stress Relaxation 59
      5.2.2 Start-up Flow 59
      5.2.3 Steady Shear Flow 61
      5.2.4 Steady Shear Free Flow 64
   5.3 Bead-Spring Tetrahedron Model with Two Constraints 66
   5.4 Multi-Bead-Spring Model with One Constraint 70

6. CONCLUSION AND DISCUSSION 72

CITED LITERATURE 77

TABLE OF CONTENTS

CHAPTER **PAGE**

APPENDICES 82
Appendix A 83
Appendix B 85
Appendix C 91
Appendix D 93
Appendix E 96
Appendix F 101
Appendix G 105
Appendix H 108

VITA 140

LIST OF TABLES

TABLE		PAGE
I.	COMPARISON BETWEEN SIMULATION METHODS	13
II.	DISTANCE BETWEEN BEADS AND NUMBER OF PARTICLES	33
III.	EXAMPLE OF STORED WORKSPACES OF 262 PARTICLE CASE	36
IV.	ALL POSSIBLE PAIR NODES AND UNIQUE PAIR NODES	37
V.	THE PARAMETERS USED FOR THE SIMULATIONS	47
VI.	COMPARISON OF ZERO-RATE VISCOSITY	58
VII.	OPTIMIZATION OF AREA SIZE	63
VIII.	STEADY STATE VISCOSITY DATA: TWO-CONSTRAINT TRIANGLE MODEL	105
IX.	SUDDEN DISPLACEMENT DATA: TWO-CONSTRAINT TRIANGLE MODEL	106
X.	ELONGATIONAL VISCOSITY DATA: TWO-CONSTRAINT TETAHEDRON MODEL	107

LIST OF FIGURES

FIGURE **PAGE**

1. (a) Schematic of a biconcave model with interconnected bead-spring triangular regions. (b) Cross-sectional diagram of RBC showing its average size. 25

2. (a) Schematic representation of the erythrocyte membrane beneath the lipid bilayer showing the protein network connection primarily in triangular grid with spectrin and actin. The nodes represent actins and the links are spectrin tetramers. (b) Representation of constraint force with simple triangular model. New arbitrarily deformed positions of the beads (solid lines) calculated based on kinetic theory from the initial configuration. Adjusted positions after constraint forces are applied and satisfied (dashed line with filled area). 26

3. The tetrahedron model with area and/or volume constraint. Total surface area is a sum of four triangles. 31

4. (1) Biconcave shape of normal human red blood cell (RBC). Modeling of RBC using mathematical expression from Kuchel (1999) in Cartesian coordinate. (2) Cross-sectional view of the 3D surface meshing model considering RBC as a thin layer of lipid bilayer sac. 32

5. Example 3D surface meshing of RBC with (a) 2006 and (b) 262 hydrodynamic resistant site. (c) Hexagons, pentagons, and septagons in the lattice.....34

6. An example of 3D surface meshing of RBC with 78 hydrodynamic resistant sites. 35

7. Flow chart of the program. 46

8. Relaxation of elastic modulus with 16.25% strain (a = 100). 50

9. Viscosity of a suspension of bead-spring triangle with one-constraint plotted verses dimensionless time for a start-up shear flow (a = 100). 51

10. Comparison of the one-constraint triangular model result against the experimental data. (a) The simulated results of a simple bead-spring-ring model with one constraint (a = 100 dimensionless unit). (b) Dintanfass (1971, 1974) used cone-in-cone rhombospheroid viscometer without the use of anticoagulant. The samples were tested immediately after withdrawal. (c) Copley and King (1973) used Weissenburg rheogoniometer with the use of dry ethylenediamine tetraacetate (EDTA) as an anticoagulant. The blood samples were that of human donors ranging in age from 25 to 60 years. (d) Windberger (2010) used Cell-Dyn 3500 and K-EDTA after the withdrawal. 52

11. A log-log plot of the first normal stress of a suspension of bead-spring triangle with one-constraint with respect to shear rate for a steady shear flow (a = 100). 53

LIST OF FIGURES

FIGURE **PAGE**

12. A comparison of result of one-constraint method with different area sizes. (a) A log-log plot of viscosity with respect to shear rate for a steady shear flow. (b) A log-log plot of first normal stress coefficient with respect to shear rate for a steady shear flow. 54

13. (a) Representation of reflection method. (b) Rejection method for capillary flow. 55

14. Fahraeus-Lindqvist effect is observed in a capillary flow. Compared with Pries (1992) analysis (a = 100). 56

15. A log-log plot of viscosity of a suspension of bead-spring triangle with one-constraint with respect to the elongation rate for a shear free flow (a = 100). 57

16. Relaxation of elastic modulus with 21.93% strain (a = 100). 59

17. Representation of how data are collected for two-constraint method to reduce computer running time. 60

18. Viscosity of a suspension of bead-spring triangle with two-constraint model plotted verses dimensionless time for a start-up shear flow (a = 100). 61

19. A comparison of result of two-constraint method against one-constraint of a triangle model (a = 100). (a) A log-log plot of viscosity with respect to shear rate for a steady shear flow. (b) A log-log plot of first normal stress coefficient with respect to shear rate for a steady shear flow. 62

20. A comparison of result of two-constraint method with different area sizes. (a) A log-log plot of viscosity with respect to shear rate for a steady shear flow. (b) A log-log plot of first normal stress coefficient with respect to shear rate for a steady shear flow. 63

21. A comparison log-log plot of elongational viscosity as a function of the elongation rate for the triangular model with constraints (a = 100). 64

22. A comparison of gyration of the model as the elongation rate increases for the triangular model with constraints (a = 100). 65

23. Comparison of viscosity plots of RBC tetrahedron model with constraint (a' = 400). 66

24. Comparison of first normal stress coefficient responses for the RBC tetrahedron model under constraint (a' = 400). 67

25. A comparison log-log plot of elongational viscosity as a function of the elongation rate for the tetrahedron model with constraints (a) a' = 100 and (b) a' = 400 68

LIST OF FIGURES

FIGURE **PAGE**

26. A comparison of gyration of the model as the elongation rate increases for the tetrahedron model with constraints (a) $a' = 100$ and (b) $a' = 400$ 69

27. A comparison of elongational viscosity of single constraint tetrahedron model with different area sizes. 70

28. A viscosity plot of the Multi-Bead-Spring RBC model with respect to shear rate ($a'' = 100$). 71

29. Gyration plot of the Multi-Bead-Spring RBC model with respect to shear rate ($a'' = 100$). 71

30. Schematic of a bending potential to the springs connected to the bead. 75

31. Schematic of using (a) center of mass of RBC (b) local center of mass to maintain the curve in the biconcave model. 75

LIST OF ABBREVATIONS

BD	Brownian Dynamics
GC	Coarse-Grained
FEM	Finite Element Method
FENE	Finite Extensible Nonlinear Elastic
FPE	Fokker-Planck equation
GNF	Generalized Newtonian Fluid
HI	Hydrodynamic interaction
MC	Monte Carlo
MD	Molecular Dynamics
ODE	Ordinary Differential Equation
RBC	Red Blood Cell
SDE	Stochastic Differential Equation

SUMMARY

In this dissertation, I present Brownian Dynamic simulation technique with constraint method to predict the movement of biological cells specifically focused on rheology of blood. Blood is often treated as continuum fluid or an empirical constitutive equation is used to study a blood flow. However, it would be impossible to observe neither the deformation nor the elasticity of the cell. The proposed method based on kinetic theory where the stress tensor and the stochastic differential equation (SDE) of motion depend on the configuration of the microstructure of the fluid will allow observing the movement as well as the material properties. In addition, the constraint method using Lagrange multiplier describes the effect of the biological cell conserving its overall size throughout the motion of flow while allowing the shape to deform. In this study, blood is considered as suspension of deformable red blood cells (RBCs) in a dilute solution of fluid. A discrete model of bead-spring RBC is constructed with linear Hookean spring to give flexibility to deform. To demonstrate the capability of the method, the minimalist bead-spring model to represent the RBC was simulated. Contraints used in this research are geometrical holonomic constraints. The RBC models are tested under shear and shear free flow. An assumption was made that the friction tensor is isotropic. The rheological material properties are obtained through simulations. A comparison is then made between predicted viscosity and experimental observations followed by discussion of the effects of constraint on each RBC models that are developed.

1. INTRODUCTION

1.1 Motivation and Purpose of the Study

In the human body as well as all other living animals, blood is indispensable to sustain life that transports essential nutrients to the tissue in the circulatory system. According to CDC (Center for Disease Control and Prevention), heart disease is the leading cause of death for people of most ethnicities in the United States. In 2006, 26% of the population died of heart disease, half of them being women. The total cost of the heart disease related health care services, medications, and lost productivity was 444 billion US dollars in 2011. Among heart related diseases, coronary artery disease is the most common type which can cause heart attack, heart failure, angina and arrhythmias. There are wide range of vascular diseases that affect circulation and other diseases in which blood rheology plays in important role such as atherosclerosis (Leschke, 2008), diabetes (Lockhart *et al.*, 2008), Crohn's disease (Novacek *et al*., 2008), sickle cell anemia (Coates, 2008), cerebral aneurysms (Valencia *et al*., 2008), and venous hypertension (Khodabandelou *et al*., 2004) just to name a few. The rheological material properties of blood are important factors in the occurrence and onset development of such diseases. The progression of disease is oftentimes accelerated by the changes in the mechanical behavior of cells. A better fundamental understanding of blood flow and its rheological properties will help rational approach to prevention of serious conditions and its costs, and development of new treatments as well.

Blood is a complex mixture of cells, proteins, lipoproteins, and ions suspended in a Newtonian fluid where erythrocyte (red blood cell, RBC) typically comprise approximately 40% of blood by volume. Studying the rheological property of RBC is a key factor of the

blood flow characteristics because of their large volume fraction. The empirical data shows that the human blood has non-Newtonian behavior and its elastic properties have been measured by Copley (1973) and Dintanfass (1974). The viscosity drops as the shear rate increases instead of remaining constant as in the Newtonian fluids. Often times the Casson model has been used to model the behavior of the blood since it gives the characteristic of negative one half power-law slope of the viscosity similar to that of experimental data. However, the model neither predicts the time-dependent behavior nor fit all the range of shear rate empirical data where viscosity tends to be Newtonian at the low and high shear limits. More importantly, the Casson model gives us no insight into the dynamics of RBC in complex flows.

The Brownian Dynamic simulation will allow predicting the movement of biological cell in methodological way with the consideration of all the forces. Instead of using an empirical constitutive equation, we can use kinetic theory where the stress tensor depends on the configuration of the fluid microstructure (*i.e*. the RBC). This will allow us to explore more about the blood flow in various conditions such as microcapillary flow, rouleaux effect or sickle cells. In this preliminary study, we propose a method to simulate a suspension of deformable microstructures. The microstructure model is red blood cell, a primary constituent of blood, suspended in a dilute solution of Newtonian fluid.

The microstructures are represented by bead-spring model where hydrodynamic resistant site referred as ‘beads’ with inter-bead potentials referred as ‘springs.’ The model is designed to capture the most essential features of RBC. A simple linear Hookean spring is used to give flexibility to deform. Constraining the size of the microstructure in the previous study has led to good prediction of mechanical properties. The crudest possible geometry, the

three bead-spring ring, with a holonomic constant-area constraint have shown that the RBC in the dilute solution deformed easily in accordance with the RBC low resistance to shear while maintaining a constant area (Lopez, 2007). It has also shown Fahraeus-Lindqvist Effect where effective viscosity is reduced in the capillary flow as RBCs squeeze through the narrow capillary tube in the microcirculatory system (Fahraeus *et al*., 1931).

There are assumptions made to the stochastic simulation of RBC. For preliminary research purpose, we assume that there is no hydrodynamic interaction (HI) between RBCs. This means that the motion of a red blood cell does not affect the position or configuration of another RBC. However, in reality, the RBCs cannot only interact, but also stack up and form rouleaux conditions which can be caused by infections, inflammatory, diabetes, tissue or disorders, and cancers, and coagulation. The rouleaux phenomena are due to the unique biconcave shape of RBC. This study is focused on vessel flow including capillary, but not considering the flow in the heart.

We would like to extend the idea of constraining method to two-constraint model. Constraints introduce difficulty that the positions of the beads are no longer independent since they are connected by the equations of the constraint. First, to test the method of multiple constraints, we tested a three bead-spring model with area and sum of length square constraints. The sum of length square constraint keeps the total length of the connected springs so that we give restriction to deformation yet still have some degree of flexibility. The flow properties of blood in different type of flow conditions are obtained. Then, the tetrahedron configuration of RBC is tested in the shear flow with area and volume constraints. Lastly, the foundational work has been done for biconcave model. With these ideas, this dissertation is organized as follows.

1.2 Organization of the Dissertation

In Section 2, background study for blood rheology is explored. The experimental data are collected from Copley (1973), Dintanfass (1974), and Windberger (2010). The constraint method by Ö ttinger (1996) and Liu's (1989) application on the constraint method is reviewed. Other literatures on constructing the RBC model and development of algorithm are discussed followed by the comparison of simulation methods.

In Section 3, the methodology that is applied for the study is explained with fundamentals of kinetic theory. Diffusion equation is constructed from the equation of motion and the equation of continuity considering all the forces applied to the system such as hydrodynamic, the Brownian, inter-particle, and constraint forces. The method of constraint that is applied to the system is discussed. Then, the development of RBC models is presented to derive the inter-particle force and apply geometric constraints. With the calculated position of the microstructure in the flow after all the forces are applied to the system, we quantify the shear stress and obtain the material functions.

In Section 4, the demonstration on how the material properties are calculated in the computer simulation and setting the parameters that are used for the simulation is discussed. In Section 5, the material properties of three-bead-spring triangular ring with one-constraint and two-constraint are compared and analyzed for start-up shear flow, steady state shear flow, capillary flow, and steady state elongational flow. For tetrahedron model, the viscosity and the first normal difference of the model in shear and shear-free flow with area and volume is illustrated. In Section 6, the summary of this research including some suggestions for the future directions of this work are discussed.

2. LITERATURE REVIEWS

First, we explore the research on related field. We will focus on work in the area of fluid models of blood, and fluid model with constraints and bead-spring model of fluid microstructure. Then, simulation techniques that are widely used for modeling of fluids such as Monte Carlo (MC) method, Molecular Dynamics (MD), and Brownian Dynamics (BD) simulations are reviewed.

2.1 Blood Rheology

The elastic properties of blood have been measured and reported by Dintenfass L. (1985), Thurston G. B (1972), Copley A. L. *et al*. (1970), Chien. Shu *et al*. (1966), Thurston G.B. *et al*. (2004), Fahraeus R. (1931), Azelvandre F., C. (1976). The Copley and King, Dintenfass, and Windberger (2010) data are used in the preliminary study to compare the results. Evans (1976) exhibits a viscoelastic response of the membrane in shear deformation in which the total shear stress is comprised of a viscous and an elastic component. Evans (1976) suggests the viscous component is due to the fluid-like behavior of the lipid bilayer whereas the elastic component is from the stretching of the cytoskeleton (Secomb, 2003). The membrane network that lies under the lipid bilayer is examined. A triangular grid is observed in the cytoskeleton where actin complexes linked by spectrin filaments (Liu S-C, 1987). To incorporate the idea of elasticity of cytoskeleton in the development of RBC model, relevant studies are reviewed.

Wiest (1987) applied kinetic theory to the ring closure structure of polymers and compared against linear polymers. Both ring closure and polymer chains were modeled as freely jointed bead-spring structure with Hookean springs. Kramers-Kirkwood expression of

bead-spring model was selected for stress tensor. It is stated in this paper that the ring closure model was first studied by Kramers with freely jointed bead-rod structure and that he obtained the zero-shear-rate viscosity comparing with Kramers chain. The zero-shear-rate viscosity is a viscosity at the lower shear rate limit where the viscosity approaches a constant value. The result showed that the ring closure structure has lowered the viscosity and first normal stress coefficient. However, there were no constraints applied to this model, therefore, the shear material properties were independent of shear rate.

The constraint method by Liu (1989) is review to apply constraint to the ring closure structure in our study. This study demonstrates sample trajectories of a dilute solution of Kramers freely jointed bead-rod chains in different flow conditions using multiple constraints method. Liu implemented the constraint method proposed by Ryckaert (1977) using iterative Lagrange multipliers procedure to constrain the length of the bond and angle. Using the Brownian dynamics simulation algorithm, the model exhibits shear thinning effects in both viscosity and first normal stress coefficient in the steady flow. In steady elongational flow, the viscosity of the solution increased drastically as the elongational rate increased. This paper gives the algorithm for BD simulations with constraints and is the basis for the simulation methods used in the work presents herein.

Fahraeus Lindqvist has done experimental work on capillary flow and found that the viscosity drops as the diameter of the vessel decreases. There have been numerical studies done for capillary flow. Secomb (2003) and Tsukada *et al*. (2001) demonstrated that the RBC shape changes gradually from biconcave shape to a parachute shape as the velocity increase in the pressure driven Poiseuille flow. This theoretical phenomenon is due to the fact that the velocity profile is parabolic and the RBC shape is deformed by the pressure effect. Evans *et al*.

(1972) used the RBC model which the biconcave shape is expressed in mathematical parametric form. The center of RBC (x_0, y_0) in the expression or the local center can be utilized to keep the biconcave shape in our study, for example.

Hosseini *et al*. (2009) used discrete 2D model where the particles are connected by non-linear springs to represent an elastic membrane. The spring force ensures conservation of the membrane area. In addition, linear bending is implemented to give the model an elasticity using resistance against deviation of the local curvature from the equilibrium curvature of the biconcave RBC shape at rest. The RBC is treated as a capsule made of an elastic membrane enclosing a Newtonian cytoplasm and is suspended in a Newtonian fluid. However, it is a continuum model that flows according to the Navier-Stokes equation. Secomb (2003) used the lubrication theory to examine the axisymmetric motion of RBC in capillary tubes, and Pozrikidis (2005) further analyzes this motion using a boundary integral method for Stokes flow where the RBC membrane is regarded as a thin shell.

Brownian dynamics simulation has been used in a number of studies of bead-spring and bead-rod model (Dotson, 1983; Atkinson, 1984; Ö ttinger, 1986; Saab, 1987; Biller, 1988; Liu, 1989). The Brownian dynamics simulations of a rouleaux effect have been investigated where aggregates of RBC stack are taken to be a dumbbell (Moyers-Gonzalez 2008). The only Brownian dynamics simulation of microstructure modeled by bead-spring chain with constraint that we are aware of is the work of Lopez (2007). Lopez reviewed different simulation techniques for computational modeling of viscoelastic fluid especially focused on Molecular dynamics (MD) and Brownian dynamics (BD). He compares constraint algorithms such as SHAKE and RATTLE (Andersen, 1983) that are widely used in MD methods to Liu's Brownian Dynamic simulation of Kramers chain. The SHAKE algorithm which is introduced

by Ryckaert works well in MD, but not BD. In Liu's method, the linear equations for the Lagrange multipliers need to be solved for each iteration step as oppose to method of SHAKE where all previous constraints are altered in order to fulfill each constraint.

Lopez's ring model with area constraint result has good agreement with empirical data showing the shear thinning effect. However, it leads to unrealistic shape of the model when the chain is subjected to large deformations. Although it is true that the shape of the microstructure in the capillary flow tends to stretch in order to squeeze through the narrow channel, the normal cell would not stretch infinitely, and therefore, the connector spring in the microstructure model should not extend infinitely. We can improve the microstructure model by adding more hydrodynamic resistant site introducing volume constraint. Tetrahedron model volume constraint can be tested under same flow condition as the triangular model. Next, Multi-Bead-Spring RBC model can be constructed and improve the simulation code to find the new configuration of the model at each time step in a systematical way. Ultimately, the goal will be to add a local force to keep the model biconcave shape in the future study.

First and foremost, we will be reviewing different simulation methodology to study blood rheology including advantages and disadvantages of the techniques.

2.2 Simulation Technique

2.2.1 Monte Carlo (MC)

A system setup for Monte Carlo (MC) statistical mechanics includes representation of molecules as collections of atom-centered interaction sites, utilization of classical force fields for the potential energy terms, and implementation of periodic boundary conditions (Jorgensen *et al*., 1996). A new configuration of the system is generated by stochastic

sampling. Acceptance of the new configuration is determined by the sampling algorithm where the application over enough configurations yields properly Boltzmann-weighted averages for structure and thermodynamic properties. In the standard Metropolis Monte Carlo, a move is accepted if the new configuration results in a lower potential energy. Or else, it is accepted with a probability given by the Boltzmann factor (Meller, 2001). As a result, average properties obtained from the accepted configurations are consistent with the canonical ensemble (NVT) where the thermodynamic state is characterized by fixed number of particles N, fixed volume V, and fixed temperature T.

The advantage of the Monte Carlo method is its generality and a relatively weak dependence on the dimensionality of the system. Finding a new configuration that would ensure efficient sampling may be a nontrivial problem. However, the ability to bias the sampling process and transition rate while retaining the essential conditions for an equilibrium ensemble provides powerful methodologies. The force bias MC method is developed to speed up relaxation in many MC systems. In some cases, basic MC method can be faster since it requires extra computation time for the calculation of the forces. With MC methods, the Helmholtz free energy of an atomic system can be obtained from an integration of the Boltzmann actor over phase space and other equilibrium properties (Gilmer *et al*., 2005).

2.2.2 Molecular Dynamics (MD)

Molecular dynamics (MD) simulation is for computing the equilibrium and transport properties of many-body system in atomistic scale. The energy of the system is calculated by discretely accounting varous inter-particle interactions such as van der Waals, covalent bonds, electrostatic force and external forces. The forces acting on particles in the system are related

to the derivative of the energy with respect to the particle position. In the classical MD, the dynamics of the system is defined by the laws of classical mechanics. The quantum MD takes the quantum nature of the chemical bond into account so that the electron density function for the valence electrons that determine bonding in the system is computed using quantum equations, whereas the dynamics of ions is followed classically (Meller, 2001).

A system setup for Molecular Dynamics is similar to Monte Carlo. The main differences are in the modes of sampling the configuration. MD is a technique to generate new configurations of the system by integration of Newton's laws of motion to all particles in the system simultaneously over a small time step to determine new positions and velocities. It explores the macroscopic properties of a system through microscopic simulations using statistical mechanics. The distribution of the system within the ensemble, collection of all possible systems which have different microscopic states but identical macroscopic or thermodynamic state, follows Boltzmann distribution. Since the sample contains a larger number of conformations, the averaged value is needed; Average values in statistical mechanics correspond to ensemble averages, and it requires integrating over all possible states of the system. Ergodic hypothesis states that the time averages equal the ensemble average allowing the system to evolve in time so that the system eventually passes through all states (Jorgensen *et al*., 1996).

MD simulation has advantage computationally over MC in the case where a system of atoms is being equilibrated at a new temperature or other change of conditions. It is more efficient due to the fact that the displacement of the particles is affected by the neighboring particles for MD so that the movement that causes a large increase in energy is rejected whereas MC generates random numbers for the unsuccessful moves. Moreover, coordinated

moves of a number of particles such as those moving into a region of reduced pressure allow fast relaxation of recovery in MD whereas such are not possible with Metropolis MC.

The time scale of the MD simulation is of the order of picoseconds (Table I). Simulations of processes on longer timescale beyond that require so many timesteps (Gilmer *et al*., 2005). Similarly, very large systems may require extensive computer resources that they cannot easily be studied by traditional all-atom methods. In these cases, instead of explicitly representing every atom of the system, reduced representations can be used. Small groups of atoms are treated as single particles which are called coarse-grained (GC) models and it increases the time and length scales. This method is widely used for membrane–protein systems.

2.2.3 Brownian Dynamics (BD)

The Brownian dynamics (BD) simulation is a powerful technique to study the structure and to simulate non-equilibrium dynamics of polymer or complex fluids in hydrodynamic flows. Explicit solvent molecules are replaced by a stochastic force taking advantage of the fact that there is a large separation in time scales between the rapid motion of solvent molecules and the slower motion of solute. It allows simulating in much larger time scales than in a molecular dynamics simulation. The stochastic differential equation is integrated forward in time to create trajectories of molecules for the study of the temporal evolution and dynamics of complex fluids such as polymers, large proteins, colloidal solutions, and so on (Doyle *et al*., 2005).

To simulate the dynamics of particles that undergo Brownian motion, force terms are added using Newton's second law; the frictional drag from the particle moving through the

viscous solvent, random collision of the solvent with the particle describing Brownian motion particle, and spring forces. Non-hydrodynamic external forces, such as magnetic or electric fields, can be added. The detailed explanation of each force is discussed further in the next chapter.

The stochastic differential equation governing the motion of the particle is called a Langevin equation. The Brownian force is taken from a random distribution that results from random interaction between a particle and the solvent molecules. Since these random events are not correlated, the average expected values of the forces are the following in order to satisfy the fluctuation-dissipation theorem.

$$\langle \boldsymbol{F}_{\nu}^{b}(t) \rangle = 0$$

$$\langle \boldsymbol{F}_{\nu}^{b}(t) \boldsymbol{F}_{\nu}^{b}(t') \rangle = 2k_B T \boldsymbol{\zeta} \delta_{ij} \delta(t - t') \boldsymbol{\delta}$$

where $\boldsymbol{F}_{\nu}^{b}$ Brownian force for particle ν, k_B is Boltzmann constant, T is absolute temperature, $\delta_{\boldsymbol{ij}}$ is Kronecker delta, $\delta(t - t')$ is Dirac delta function, and $\boldsymbol{\delta}$ is unit tensor. These descriptions are equivalent to that of Fokker-Planck equation which we will go in depth in the main method part of this dissertation. Fokker-Planck equation (FPE) is a diffusion equation for the phase space probability density function $F(\boldsymbol{r}_{\boldsymbol{\nu}}, \dot{\boldsymbol{r}}_{\boldsymbol{\nu}}, t)$. This approach solves the FPE directly for $F(\boldsymbol{r}_{\nu}, \dot{\boldsymbol{r}}_{\nu}, t)$ whereas, in the Langevin approach, the phase space trajectories are found from the strict Langevin equation and $F(\boldsymbol{r}_{\nu}, \dot{\boldsymbol{r}}_{\nu}, t)$ can be obtained by averaging over the trajectories. When the Maxwellian velocity distribution assumption is used to reduce the FPE into an equation for the coordinate space distribution function, the equation is referred as the diffusion equation (Doyle *et al*., 2005).

A summary of simulation methods are compared in the Table I below.

TABLE I
COMPARISON BETWEEN SIMULATION METHODS

Method	Advantages	Disadvantages
Monte Carlo	Atomic-level Large scale sampling Useful statistics	Difficult to devise structural perturbations
Molecular Dynamics	Continuous motion Microscopic level Experimental bridge between structures and macroscopic kinetic data Equilibrium	Short time span
Brownian Dynamics	Mesoscopic Larger time scales Non-equilibrium state	Limited to systems with small inertia

3. METHODOLOGY

The BD simulation has advantage over other simulation techniques we have reviewed in the previous section for the study of blood flow. The recent study of molecular dynamics incorporates randomness to its study which is equivalent to Brownian motion; however, we use kinetic theory since explicit structure of solvent is not in our interest. The kinetic theory has advantages over the Monte Carlo method because it can also consider non-equilibrium state. In this dissertation, we use kinetic theory to explore rheology of blood with a discrete model of RBC where the Brownian random motion describes the movement of RBC particles. Simulating dilute solution of RBC in simple linear flows will be a benchmark to evaluate the ability of BD simulations on biological systems with mesoscopic size of the cell.

We first start with an explanation of all the possible forces that act on the beads. Then the diffusion equation is derived from the equation of motion and the equation of continuity. In addition to the Brownian motion, we apply constraint method to restrict the motion so that the RBC microstructure moves and stretch in more plausible way in the flow. The information we can collect from this study is stress tensors in which we can define the rheological properties to analyze the characteristic of the fluid. Also, we can investigate the configuration and distribution of RBCs in more complex flows.

3.1 Kinetic Theory: Diffusion Equation

3.1.1 The Equation of Motion

In this research, blood is treated as red blood cells (RBC) suspended in a Newtonian fluid. The RBC is constructed with spherical beads and each bead is subject to be influenced by a

variety of forces. The main forces under consideration are a hydrodynamic force in the form of a Stokes' law drag, the Brownian force, inter-particle forces, and constraint forces. In general, the equations of motion for each bead of the RBC can be represented by the sum of all the forces acting on the bead.

$$\boldsymbol{F}_\nu^h + \boldsymbol{F}_\nu^b + \boldsymbol{F}_\nu^\phi + \boldsymbol{G}_\nu = m_\nu \ddot{\boldsymbol{r}}_\nu = 0 \qquad \nu = 1, 2, 3, \dots, N \qquad (3.1)$$

The indices ν denote the number of particles throughout this study. The mass m_ν of each bead is assumed identical for all beads and $\boldsymbol{r}_\nu$ is position vector. The inertial term $m_\nu \ddot{\boldsymbol{r}}_\nu$ is neglected assuming that the inertia of the beads is negligible as they move through the viscous medium. This assumption of low Reynolds number has been tested for polymer suspension (Cordoba, 2012); however, in the case of RBCs, inertia may have a greater effect, but we have ignored it as a justifiable simplification. The effects of bead inertia on the Rouse model are found by solving coupled ordinary differential equations (ODE) involving the ensemble average (Schieber, 1988). The bead-spring chain model is called the Rouse and when the hydrodynamic interaction is included it is referred as the Zimm model. In the following subsequence sections, each of the contribution to the force balance is described.

(a) The hydrodynamic drag force

The hydrodynamic drag force is a resistance that the bead experiences as it moves through the viscous solution being influenced by the motion of other bead as well. The velocity of homogeneous flow field at bead ν is $\boldsymbol{v}_\nu = \boldsymbol{v}_0 + [\boldsymbol{\kappa} \cdot \boldsymbol{r}_\nu]$ ($\boldsymbol{\kappa}$ is sum of the velocity gradient and its transpose $\nabla \boldsymbol{v}^\dagger$, see Appendix E), and $\boldsymbol{v}_\nu'$ is the perturbation of the flow field at bead ν that is from the motion of other bead. This perturbation is called the hydrodynamic interaction. Then the hydrodynamic drag force can be expressed so that the force is proportional to the

difference between the averaged bead velocity $\dot{\boldsymbol{r}}_\nu$ and the velocity $(\boldsymbol{v}_\nu + \boldsymbol{v}'_\nu)$of the solution at bead ν. The definition of averaged value can be found in Appendix A.

$$\boldsymbol{F}_\nu^h = -\boldsymbol{\zeta} \cdot [[\![\dot{\boldsymbol{r}}_\nu]\!] - (\boldsymbol{v}_\nu + \boldsymbol{v}'_\nu)] \tag{3.2}$$

The friction tensor $\boldsymbol{\zeta}$ is assumed to be isotropic so that $\boldsymbol{\zeta} = \zeta\boldsymbol{\delta}$ where the scalar $\zeta$is the friction coefficient. Note that $[\![\dot{\boldsymbol{r}}_\nu]\!]$is the momentum-space-averaged velocity of the bead; It is not evaluated by Maxwell velocity distribution since it gives the fluid velocity of $\boldsymbol{v}$ and not necessary to evaluate explicitly as this is substituted into the equation of continuity (Bird *et al.*, 1987). We neglect the term $\boldsymbol{v}'_\nu$ that is from the motion of other bead.

(b) The Brownian force

A particle with small mass tends to move randomly making the movement of the bead an irregular path because of the thermal fluctuations in the liquid. The average of this rapidly and irregularly fluctuating force can be expressed with the configurational distribution function $\Psi(\boldsymbol{r}_1, \boldsymbol{r}_2, \dots, \boldsymbol{r}_N, \mathrm{t})$. The standard expression for the Brownian force is then given by

$$\boldsymbol{F}_\nu^b = -\frac{1}{\Psi}\frac{\partial}{\partial \boldsymbol{r}_\nu} \cdot [[\![m(\dot{\boldsymbol{r}}_\nu - \boldsymbol{v})(\dot{\boldsymbol{r}}_\nu - \boldsymbol{v})]\!]\Psi] \tag{3.3}$$

When the equilibration in momentum space is assumed, average velocity space distribution function can be evaluated using Eq. A-4 in Appendix A. That is assuming the velocity distribution to be Maxwellian about the fluid velocity at the center-of-mass of the polymer model. Then the contribution of Brownian motion can be simplified as

$$\boldsymbol{F}_\nu^b = -k_B T \frac{\partial \ln \Psi}{\partial \boldsymbol{r}_\nu} \tag{3.3a}$$

where k_B is the Boltzmann constant and T is absolute temperature. Thus, the $k_B T$ factor is from the velocity of the bead due to thermal fluctuation.

(c) The inter-particle force

The inter-particle force on a bead results from the connected springs in the model given as the negative gradient of the spring potential energy ϕ as following.

$$\boldsymbol{F}_{\nu}^{\phi} = -\frac{\partial \phi}{\partial \boldsymbol{r}_{\nu}} \tag{3.4}$$

Different types of springs can be used for the modeling of suspended microstructures in a fluid. Defining connector force $\boldsymbol{F}_{\nu}^{c}$ and acquiring appropriate inter-particle force for specific model is discussed in detail in Section 3.3.1 and Section 6.

(d) The constraint force

The contribution of constraint force on bead ν is expressed as

$$\boldsymbol{G}_{\nu} = -\sum_{j=1}^{d'} \lambda_j(t) \frac{\partial \sigma_j}{\partial \boldsymbol{r}_{\nu}} \tag{3.5}$$

where σ is the given constraint on the system and d' is number of applied constraints (Öttinger, 1996). Further in-depth discussion is in Section 3.2.

The hydrodynamic forces tend to distort the microstructure as oppose to inter-particle forces which tend to restore the microstructure to its original shape. The Brownian forces randomize the orientation of the microstructure. In the case of dilute suspension of RBC, we do not expect the Brownian force to significantly affect the center-of-mass diffusion of the cell. It rather causes fluctuation in the cell's shape, and therefore, influences the stress in the cell. Some researchers have ignored Brownian forces because the Péclet number for the center-of-mass motion of the RBC is above the colloidal limit. However, it is not clear that the effects of Brownian fluctuations on the shape of the RBC are insignificant and there will

not have significant effect on the viscosity. Indeed, lipid membranes are extremely soft and are easily deformed by thermal fluctuations. This has long been known experimentally as the flicker phenomenon (Blowers, 1951 and Brochard, 1975), caused by RBC membrane undulations. Therefore, we choose to include Brownian effects in our model.

Although beads of the particle can perturb the flow field in the neighboring particles, hydrodynamic interaction is excluded in preliminary study. The interactions between beads would be expected to be important in concentrated systems. In this study, we mainly consider inter-particle, Brownian, and constraint forces that gives restriction to the movement of particle. Lopez (2007) studied a RBC model in which only a single constraint was used. In this study we will examine a two-constraint method. When Eq. 3.1 is numerically integrated, the dependence of all the forces on beads has to be obtained from the relations of constraint. Although the constraints are fulfilled at the beginning, integrating the equation of motion with constraint force will cause a discrepancy in the constraints at the end of time integration due to the approximate character of the numerical calculations. In order to make all the constraints satisfy at each time step of the integration, we can first integrate Eq. 3.1 without the constraint force and obtain the position of the beads. Then, we can apply constraints which will be discussed in next section in detail.

Substituting Eqs. 3.2, 3.3a, 3.4 into 3.1 gives

$$-\zeta[[\dot{\boldsymbol{r}}_\nu]] - (\boldsymbol{v}_0 + [\boldsymbol{\kappa}\cdot\boldsymbol{r}_\nu])] - k_B T \frac{\partial \ln \Psi}{\partial \boldsymbol{r}_\nu} + \boldsymbol{F}_\nu^\phi = 0 \qquad (3.6)$$

In the next section, the equation of continuity is combined with Eq. 3.6 to give the differential equation, which is the basis for the Brownian dynamics simulation method.

3.1.2 The Equation of Continuity

In equilibrium systems, the expression configurational distribution function can be directly obtained by equilibrium statistical mechanics. For non-equilibrium system, we can derive a second-order partial differential equation for the configuration-space distribution function, so called the diffusion equation, by combining the force balances on the beads (Eq. 3.6) with the equation of continuity in configuration space. Bird *et al.* (1987) gives a thorough explanation of the principle of kinetic theory and C. F. Curtiss derived the diffusion equation for general bead-rod-spring models of dilute solutions (Bird, 1987).

Considering the time rate of change of system points (the location and orientation of beads in the microstructure model) within a hypercube, the equation of continuity for Ψ shows the conservation of system points.

$$\frac{\partial \Psi}{\partial t} = -\sum_{\nu}\left(\frac{\partial}{\partial \boldsymbol{r}_{\nu}} \cdot [\![\dot{\boldsymbol{r}}_{\nu}]\!]\Psi\right) \tag{3.7}$$

$$\frac{\partial \Psi}{\partial t} = -\left(\frac{\partial}{\partial \boldsymbol{r}_{c}} \cdot [\![\dot{\boldsymbol{r}}_{c}]\!]\Psi\right) - \sum_{j}\left(\frac{\partial}{\partial \boldsymbol{Q}_{j}} \cdot [\![\dot{\boldsymbol{Q}}_{j}]\!]\Psi\right) \tag{3.7a}$$

The Eq. 3.7 can be expressed in the form of connector vectors $\boldsymbol{Q}_{\nu}$ and the center-of-mass $\boldsymbol{r}_{c} = \frac{1}{\mathrm{N}}\sum_{\nu}^{\mathrm{N}} \boldsymbol{r}_{\nu}$ as shown in Eq. 3.7a by substituting the expression $[\![\dot{\boldsymbol{r}}_{\nu}]\!]$ with $[\![\dot{\boldsymbol{Q}}_{j}]\!]$ and $[\![\dot{\boldsymbol{r}}_{c}]\!]$. The expression $[\![\dot{\boldsymbol{r}}_{c}]\!]$ can be obtained by adding ν number of equations in Eq. 3.6 and then divide by N. When these equations are subtracted using the relations in Eq. A-2 (Appendix A) and the chain rule of partial differential equations, we obtain the equation of motion for the connector vectors $\boldsymbol{Q}_{\nu}$. Connector vectors are defined in Appendix B for each model. In this study, the Cartesian coordinate is chosen so that we obtain the diffusion equation in terms of position of the beads by simply substituting $[\![\dot{\boldsymbol{r}}_{\nu}]\!]$ using Eq. 3.6 into Eq. 3.7.

$$\frac{\partial \Psi}{\partial t} = -\sum_{\nu=1}^{N} \frac{\partial}{\partial \boldsymbol{r}_\nu} \cdot \left[\left(\boldsymbol{v}_\nu + \frac{1}{\zeta} \boldsymbol{F}_\nu^{\boldsymbol{\phi}} \right) \Psi \right] + \frac{k_B T}{\zeta} \sum_{\nu=1}^{N} \frac{\partial}{\partial \boldsymbol{r}_\nu} \cdot \frac{\partial}{\partial \boldsymbol{r}_\nu} \Psi \qquad (3.8)$$

We are able to recognize that Eq. 3.8 is a form of well-known Fokker-Planck (FPE or Kolmogorov forward) equation where the general form is

$$\frac{\partial f}{\partial t} = -\sum_{i=1}^{N} \frac{\partial}{\partial x_i} \left[D_{1_i}(x_1, \dots, x_N) f \right] + \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial^2}{\partial x_i \partial x_j} \left[D_{2_{ij}}(x_1, \dots, x_N) f \right] \qquad (3.9)$$

with Ito drift vector term $D_{1_i}(x,t)$ and diffusion tensor term of $D_{2_{ij}}(x,t)$. This second order partial differential equation describes how the system points diffuse in the multidimensional configuration space.

When we consider the Itô SDE, the relationship between Fokker-Planck equation and stochastic differential equation (SDE) is

$$dX_t = \mu(X_t, t)dt + \epsilon(X_t, t)dW_t \qquad (3.10)$$

An independent Wiener process W_t that is generated by the SDE is a three dimensional Gaussian white noise process caused by the random Brownian force.

The drift and diffusion term in the Fokker-Planck equation are

$$D_{1_i}(x,t) = \mu_i(x,t) \qquad (3.11a)$$

$$D_{2_{ij}}(x,t) = \frac{1}{2} \sum_k \epsilon_{ik}(x,t) \epsilon_{jk}(x,t) \qquad (3.11b)$$

Then the Eq. 3.10 becomes

$$d\boldsymbol{r}_\nu(t) = \left(\boldsymbol{v}_\nu + \frac{1}{\zeta} \boldsymbol{F}_\nu^{\boldsymbol{\phi}} \right) dt + \sqrt{\frac{2k_B T}{\zeta}} dW_\nu(t) \qquad (3.12)$$

This also agrees with Risken (1989) and Ö ttinger (1996) with the assumption of isotropic friction tensor, and no hydrodynamic interaction.

The Euler scheme for integration is applied to get the positions of the beads at time$(t + \Delta t)$.

$$\boldsymbol{r}_{\nu}^{UN}(t + \Delta t) = \boldsymbol{r}_{\nu}(t) + \left[\left(\boldsymbol{v}_{\nu} + \frac{1}{\zeta}\boldsymbol{F}_{\nu}^{\boldsymbol{\phi}}\right)\right]\Delta t + \sqrt{\frac{2k_BT}{\zeta}}\Delta W_{\nu} \qquad (3.13)$$

where $\Delta W_{\nu} = W_{\nu}(t + \Delta t) - W_{\nu}(t)$ is Gaussian white noise.

Once the displacement of beads in the RBC model is calculated from the stepwise integration (Euler scheme solution) of SDE, constraints can be applied to adjust positions of the beads. This adjustment is made for each time step by Lagrange multiplier to enforce the constraint. Averaged rheological material properties of the RBC solution can be calculated accounting all the number of trajectories. The calculation of the material properties depend on the type of flow and will be discussed in Section 3.5.

The positions of the beads by applying constraint force using Lagrangian method is further discussed in the following section.

3.2 Method of Constraint with Lagrange Multiplier

A constraint can be applied to a system in order to give a condition to satisfy. For example, the constraint can be a length of the bond or an angle between the bonds since the bonds between the elements maintain particular angle in reality due to the attraction and repulsion of electrons in its orbitals. The constraint algorithm where the equations of motion are solved while simultaneously satisfying all the constraints at each step of the integration was developed for the use in molecular dynamics (Ryckaert, 1977): however, it can be implemented to Brownian dynamics. Liu (1989) used Brownian dynamics simulation to

calculate rheological properties of Kramers freely jointed bead-rod polymer chain with constant lengths and angles (Ö ttinger, 1996).

Here, we will apply geometrical constraints to the erythrocyte so that it conserves its overall size throughout the motion of flow in the blood while allowing the shape to change. This constraint will strongly resist the constructed model from stretching infinitely which will not happen in reality. It is possible to select the springs that are connected to the beads other than Hookean in order to maintain the length of springs in microstructure. The magnitude of the end-to-end vector of the linear Hookean chain model has no upper bound that it can in fact extend to infinity. Polymer molecule, in general, has a finite fully extended length. For such reason, there are improved spring models such as FENE (finitely extensible nonlinear elastic) or Fraenkel spring which the force law between the beads of the chain has modified so that the chain stiffens as its extension increases. However, FENE has singularity and simple linear Hookean is selected to give more degree of freedom to stretch knowing the fact that the RBC in the micro vessel can lengthen its shape more than in the other flow conditions.

The result of three-bead-spring ring RBC model with one constraint was reported by Lopez (2007). We will observe whether the multiple constraints method complies with preliminary model using two constraints. Depending on how we set up the constraints, number of bead positions involved will be different. Therefore, calculating the gradient can be complicated as more beads are considered for each constraint. It is our interest to find a pattern to set up the constraint and calculate its derivative in a systematical way for future development of a biconcave model.

A system of N particles with d' holonomic constraints has $3N - d'$degrees of freedom in Cartesian coordinate. The positions of the beads are connected by the equations of

constraints such that the constraints limit the motion of the particles in the system. The form of holonomic constraints are given below.

$$\sigma_j(\{\boldsymbol{r}_\nu\}) = f(\boldsymbol{r}_1, \boldsymbol{r}_2, \dots, \boldsymbol{r}_N, t) = 0 \qquad \text{for } j = 1, 2, 3, \dots, d' \quad (3.14)$$

where d' is the number of constraint equations and $\boldsymbol{r}_\nu$ are the coordinates of N particles with indices of $\nu = 1, 2, 3, \dots, N$. The holonomic constraints for preliminary and proposed model are set in Section 3.3.1 and Section 6. The derivative of the constraint is in Appendix B.

There are various approaches to apply constraints to the system. It can be done at different level; Fokker-Planck equation, SDEs, or numerical integration scheme. Although it should be theoretically equivalent, there can be some deviation in these approaches depending on how strictly the constraint is satisfied since they are approximation schemes. The classification of different approaches and their advantages and disadvantages are discussed in Ö ttinger (1996). The numerical integration scheme in Cartesian coordinates is used here to solve SDE.

The total contribution of constraint $\boldsymbol{G}_\nu$ on bead ν was defined in Eq. 3.5. The time-dependent Lagrange multipliers λ_j that are associated with σ_j are determined to satisfy Eq. 3.14 at each time step. The Lagrange multipliers method is used for optimization to find the maximum or minimum of a function subject to constraints. The Lagrange formulation has advantage of giving a scalar value as opposed to vector.

From the system without constraint in Eq. 3.13, we can optimize the positions by applying constraint force $\boldsymbol{G}_\nu$ using Lagrangian method to get the constrained positions of the beads at time $t + \Delta t$.

$$\boldsymbol{r}_\nu^{CON}(t + \Delta t) = \boldsymbol{r}_\nu^{UN}(t + \Delta t) - \frac{1}{\zeta}\sum_{j=1}^{d'} \lambda_j \left[\nabla_\nu \sigma_j\right]_c \qquad (3.15)$$

$$\lambda_j = \sum_{k=1}^{d'} \left[\overline{g}_{jk}\right]_{c'} \sigma_k \qquad (3.16)$$

At each step of integration in Eq. 3.12, the iterative method for constraint is treated individually. The superscript *CON* denotes constrained and *UN* denotes unconstrained. The unconstrained position starts with Eq. 3.13, and then the constraint converges by iteration until the second term in Eq. 3.15 approaches zero. This means the estimation of the positions of the beads satisfies the constraint that we set up and the correction is no longer needed.

There are $N + d'$ equations in 3.14 and 3.15 that need to be solved simultaneously for $N + d'$ variables. The unknown variables are $\boldsymbol{r}_1^{CON}, \boldsymbol{r}_2^{CON}, \ldots, \boldsymbol{r}_N^{CON}$ and $\lambda_1, \lambda_2, \ldots, \lambda_{d'}$.

The bracket $[\ldots]_{c'}$ means the term is evaluated at $c' \in [0,1]$ with zero being the old position at t and one being the new position at $t + \Delta t$ (Ö ttinger, 1996). In the simulation code, $c = 0$ and $c' = 1$ is used. Choosing which position to determine Lagrange multiplier is a matter of number of iteration in the constraint subroutine which also depends on how the constraint is set up. Overall, for the case where Lagrange multiplier is calculated only with the old position, the computation is less expensive in a sense that it does not need to recalculate the derivative of new position for every time step to satisfy the constraint. However, it converge better using new position as the model structure gets complicated with multi-beads. In addition, it is better to avoid setting an initial position exactly to equilateral triangle. In that case, the constraint did not converge because it gives the denominator of the Lagrange multiplier to be practically zero. Analytically, determinant of a modified metric matrix is not zero as proved in Appendix D.

The metric matrix $\overline{g}_{jk}$ is an inverse of modified metric matrix $\overline{G}_{jk}$, therefore, it satisfies

$$\sum_{l=1}^{d'} \overline{g}_{jl} \overline{G}_{lk} = \delta_{jk} \qquad (3.17)$$

Modified metric matrix with dimension $d' \times d'$is

$$\overline{G}_{jk} = \frac{1}{\zeta}\sum_{\nu=1}^{N} \frac{\partial \sigma_j}{\partial \boldsymbol{r}_\nu} \cdot \frac{\partial \sigma_k}{\partial \boldsymbol{r}_\nu} \qquad (3.18)$$

assuming the friction tensor $\boldsymbol{\zeta}$ is isotropic and there is no hydrodynamic interaction. One other way to determine the Lagrange multiplier is from Taylor expansion of the constraint as shown in Lopez (2007). Detailed calculations of Lagrange multiplier for one and two constraints are demonstrated in Appendix C.

Next, we will examine how the configuration of the RBC model is constructed and propose a new model with volume. Moreover, constraints will be set up for particular model to apply Lagrangian method.

3.3 Model Development

To construct a simulation model of RBC, we can examine structure of the cell first. For a healthy normal human, RBC is a flexible biconcave disk with approximately 8 μm(microns) in diameter and cell thickness is about 2 microns. It is a lipid bilayer vesicle with volume of approximately 90 fL containing a cytoplasm and hemoglobin. The thickness of bilayer is approximately 5 nm which is roughly 1/1000 of its size of cell. Its membrane is highly elastic and deformable.

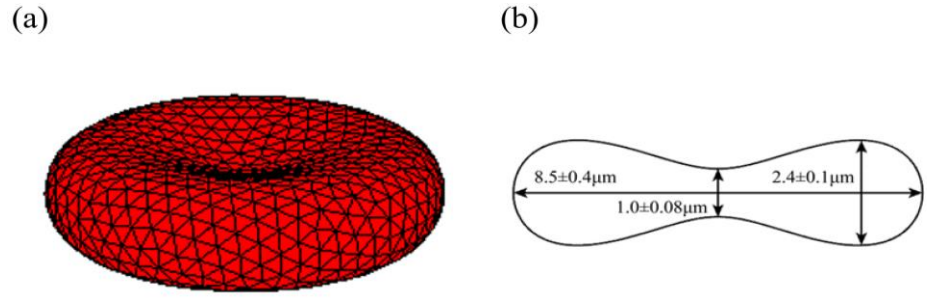


Figure 1. (a) Schematic of a biconcave model with interconnected bead-spring triangular regions. (b) Cross-sectional diagram of RBC showing its average size.

Underneath the lipid layer, there is protein network primarily in triangular grid (Figure 1). The membrane structure of human erythrocytes was examined by Liu S-C (1987) using high resolution negative staining electron microscopy. The study shows that this cytoskeleton of RBC is triangular grid configuration with mostly hexagonal lattice and the rest which is approximately 11% of lattice structure is shown as pentagons and septagons. Each triangular grid observes actin complexes linked by spectrin filaments. The end-to-end distance of spectrin tetramer is approximately 100 nm between the nodes and the total length when it is stretched is 200 nm (Liu S-C, 1987).

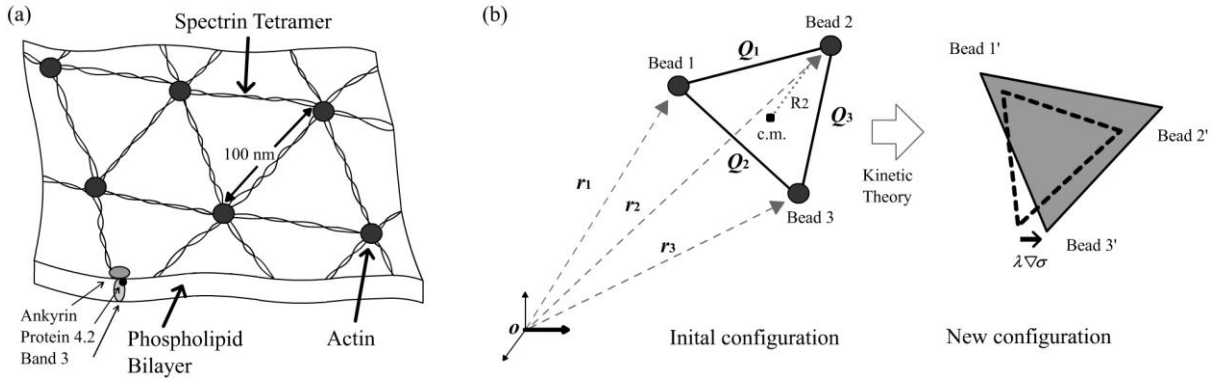


Figure 2. (a) Schematic representation of the erythrocyte membrane beneath the lipid bilayer showing the protein network connection primarily in triangular grid with spectrin and actin. The nodes represent actins and the links are spectrin tetramers. (b) Representation of constraint force with simple triangular model. New arbitrarily deformed positions of the beads (solid lines) calculated based on kinetic theory from the initial configuration. Adjusted positions after constraint forces are applied and satisfied (dashed line with filled area).

When constructing the kinetic theory models, beads, springs, and rods are the typical building blocks. Incorporating the structure of cytoskeleton to the model development, the structure of RBC can be modeled as a collection of beads connected with springs with triangular grid to represent the membrane of the cell. All the beads are assumed to have identical mass. The spring selection is not limited to, but the Hookean spring is selected as a connector to demonstrate the deformation of the RBC. It is a simple linear spring model that

gives a linear relation between the tension and separation of any given masses that are joined together. With this spring connection, the RBC model can capture flexibility and elasticity of the membrane. Moreover, by selecting Hookean spring, it can avoid overlap effect of the constraint.

The model can have degrees of complexity in order to have good estimate of the RBC configuration. In this dissertation, three models are tested. We first start out with very crude triangular model with constraint to roughly see how macroscopic fluid behavior is related to molecular orientation and stretching. It proves that the method using kinetic theory works for non-linear polymer structure with a minimum of mathematics. Two combined constraint has been applied to the model to test multiple constraint method. Then, tetrahedron model is constructed to test volume constraint which eventually would be applied to biconcave model. Finally, the foundation for biconcave model is developed in this dissertation.

For such model, the potential forces on the beads are related by tension in the spring and the connection between beads and springs. Thus, as seen in Section 3.1, the equation of motion which results in the stochastic differential equation (SDE) of motion depends on the RBC configuration. In the following subsections, RBC configuration for each model is constructed, connectors are defined, expressions for inter-particle forces are found, and applied constraints are setup.

3.3.1 Inter-particle Forces and Constraint Setup for Each Models

The Three-Bead-Spring Ring Model with Two Constraints

The simplest triangular bead-spring RBC model with Hookean spring and area constraint was proposed by Lopez (2007). This is a ring structure with three identical beads, each connected to two adjacent springs. Further study of this model is done to observe the effect of multiple constraint method described in Section 3.2. Additional sum of length square constraint is applied to the system. As in the previous study (Lopez, 2007), the polymer kinetic theory is used for the triangular model with two constraints. The hypothesis is that the constraints of RBC are yet preserved while RBC deforms.

Without any constraint, the triangular model becomes a Rouse model with ring closure effect (Rouse, 1953; Wiest, 1986). Lopez (2007) presented the inter-particle force by using the definition for ring structures. The connector vector for the ring closure structure is

$$\boldsymbol{Q}_k = \boldsymbol{r}_{k+1} - \boldsymbol{r}_k \tag{3.19}$$

where

$$\sum_{k}^{N} \boldsymbol{Q}_k = 0$$

so that the connector vector for *N-th* bead is $\boldsymbol{Q}_N = \boldsymbol{r}_1 - \boldsymbol{r}_N$. The position vectors $\boldsymbol{r}_\nu$ denote the absolute positions of the beads with respect to the fixed reference frame in space as shown in Figure 2(b). The force acting on each bead through two adjacent springs has the following relation between the potential forces on the beads $\boldsymbol{F}_\nu^\phi$ and the tension in the spring $\boldsymbol{F}_\nu^c$ using Eq. 3.4.

$$\boldsymbol{F}_\nu^\phi = -\sum_{k}^{N-1}\left(\bar{B}_{k\nu} - \delta_{1,\nu} + \delta_{N,\nu}\right)\boldsymbol{F}_k^c \tag{3.20}$$

with

$$\bar{B}_{k\nu} = \delta_{k+1,\nu} - \delta_{k,\nu}$$

$$\boldsymbol{F}_k^c = H\boldsymbol{Q}_k$$

When substituted, the forces can be expressed with the connector vectors which are relative positions that do not depend on the reference frame. The parameter *H* is a constant for Hookean spring and the spring force for different types of spring is well explained in Bird *et al.* (1987). Time constant for Hookean springs for the model is expressed as λ_N. For Example, this value is $\lambda_H = \zeta/4H$ for the Hookean dumbbell model and $\lambda_N = \sum_k^{N-1} \lambda_k = \frac{\zeta}{4H}\left[\frac{N^2-1}{3}\right] = \lambda_H\left[\frac{N^2-1}{3}\right]$ for Rouse chain, and $\lambda_N = \sum_k^{N-1} \lambda_k = \frac{\zeta}{4H}\left[\frac{N^2-1}{6}\right]$ for ring closure.

We can generalize that

$$\lambda_N = \frac{\zeta}{4H} f(\mathrm{N}) = \lambda_H f(\mathrm{N}) \qquad (3.21)$$

Units of time, length, and mass used in the simulation is such that $\lambda_N/f(\mathrm{N}) = 1$, $k_BT/H = 1$, $n_c k_B T = 1$.

Equation 3.20, however, only applies to the ring closure structure. Knowing that two connector vectors are going to be used to calculate the area of the triangle and the pattern for the relation between potential forces and the tension has to be developed as the number of beads increase, the connector vectors for the triangular structure in this study is defined as following for convenience.

$$\boldsymbol{Q}_1 = \boldsymbol{r}_2 - \boldsymbol{r}_1, \quad \boldsymbol{Q}_2 = \boldsymbol{r}_3 - \boldsymbol{r}_1, \quad \boldsymbol{Q}_3 = \boldsymbol{r}_3 - \boldsymbol{r}_2 \qquad (3.22)$$

Then, the inter-particle forces on each bead are

$$F_1^{\phi} = H(\boldsymbol{Q}_1 + \boldsymbol{Q}_2) \qquad (3.23a)$$

$$F_2^{\phi} = H(\boldsymbol{Q}_3 - \boldsymbol{Q}_1) \qquad (3.23b)$$

$$F_3^{\phi} = -H(\boldsymbol{Q}_2 + \boldsymbol{Q}_3) \tag{3.23c}$$

for a triangular model.

For the area constraint, an area of a triangle can be calculated with the cross product of two vectors. Since the cross product of two vectors is a vector, it can be dotted by itself which is also the square of the magnitude of the vector so that the constraint gives scalar value back and it is simpler to obtain its derivative. Then, the imposed constraint of constant area for each ring is

$$\begin{aligned}\sigma_1(\{\boldsymbol{r}\}) &= \sum(\text{area})^2 - \text{constant} \\ &= \left\{\tfrac{1}{2}|\boldsymbol{Q}_1 \times \boldsymbol{Q}_2|\right\}^2 - a = 0\end{aligned} \tag{3.24}$$

in which a is a scalar value of constant area.

Similarly, the second constraint can be sum of length square constraint and postulated as following.

$$\sigma_2(\{\boldsymbol{r}\}) = (|\boldsymbol{Q}_1|^2 + |\boldsymbol{Q}_2|^2 + |\boldsymbol{Q}_3|^2) - l = 0 \tag{3.25}$$

The constraint is set to be a sum of square of the length of each spring rather than perimeter for simpler form of its derivative. The derivatives of each constraint are in Appendix B. The constants l are lengths of sides of a triangle.

Three-Bead-Spring Ring Tetrahedron Model with Two Constraints

As the healthy RBC flows through the vessel in the body, it will try to conserve its shape. Maintaining shape can be represented as having constant surface area and volume. The two-constraint method has been tested with the triangle model in previous section. We can extend

the model to have volume by adding one additional bead and three additional springs to make tetrahedron as shown in the Figure 3.

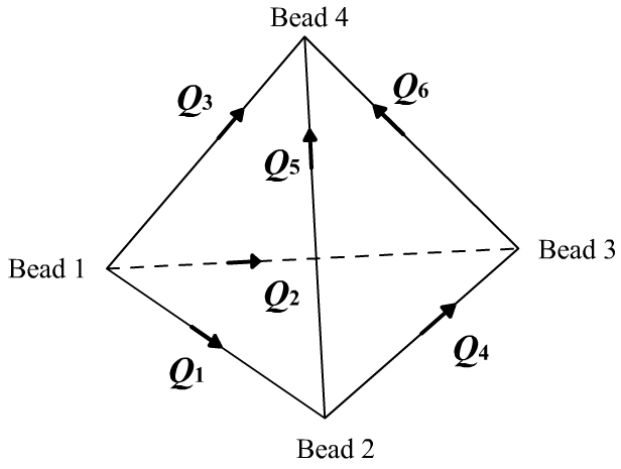


Figure 3. The tetrahedron model with area and/or volume constraint. Total surface area is a sum of four triangles.

As in preliminary study, same assumption is made that all the beads have identical mass. Each bead is connected to three adjacent springs. The connector vectors are defined as

$$\boldsymbol{Q}_1 = \boldsymbol{r}_2 - \boldsymbol{r}_1, \quad \boldsymbol{Q}_2 = \boldsymbol{r}_3 - \boldsymbol{r}_1, \quad \boldsymbol{Q}_3 = \boldsymbol{r}_4 - \boldsymbol{r}_1, \tag{3.26}$$

$$\boldsymbol{Q}_4 = \boldsymbol{r}_3 - \boldsymbol{r}_2, \quad \boldsymbol{Q}_5 = \boldsymbol{r}_4 - \boldsymbol{r}_2, \quad \boldsymbol{Q}_6 = \boldsymbol{r}_4 - \boldsymbol{r}_3$$

In similar way as it is done for triangular model, the potential forces for tetrahedron are:

$$F_1^{\phi} = H(\boldsymbol{Q}_1 + \boldsymbol{Q}_2 + \boldsymbol{Q}_3) \tag{3.27a}$$

$$F_2^{\phi} = H(\boldsymbol{Q}_4 + \boldsymbol{Q}_5 - \boldsymbol{Q}_1) \tag{3.27b}$$

$$F_3^{\phi} = H(\boldsymbol{Q}_6 - \boldsymbol{Q}_2 - \boldsymbol{Q}_4) \tag{3.27c}$$

$$F_4^{\phi} = -H(\boldsymbol{Q}_3 + \boldsymbol{Q}_5 + \boldsymbol{Q}_6) \tag{3.27d}$$

Now, we set up constraints for tetrahedron model. The area constraint is

$$\sigma_1(\{\boldsymbol{r}\}) = \frac{1}{4}\begin{pmatrix} |\boldsymbol{Q}_1 \times \boldsymbol{Q}_2|^2 + |\boldsymbol{Q}_2 \times \boldsymbol{Q}_3|^2 \\ +|\boldsymbol{Q}_3 \times \boldsymbol{Q}_1|^2 + |\boldsymbol{Q}_4 \times \boldsymbol{Q}_5|^2 \end{pmatrix} - a' = 0 \tag{3.28}$$

where $a'$is a constant for total surface area which is a value for sum of square of the area of each triangle.

The volume constraint is set using the fact that $1/6$ of the triple product represent the volume

$$\sigma_2(\{\boldsymbol{r}\}) = \frac{1}{6}(\boldsymbol{Q}_3 \cdot [\boldsymbol{Q}_1 \times \boldsymbol{Q}_2]) - b = 0 \qquad (3.29)$$

where b is a constant for volume. See Appendix B for the calculation of derivatives for each constraint.

Multi-Bead-Spring 3D Mesh Model with Total Surface Area Constraint

Further study on more complicated geometric modeling of RBCs as in Figure 4 is done. In this section, a systematical way to find neighboring beads is developed to calculate inter-particle forces. The initial biconcave shape of RBC is set using mathematical expression by Kuchel (1999). Detail regarding the expression is in the MATLAB function 'discocyte' in Appendix F.

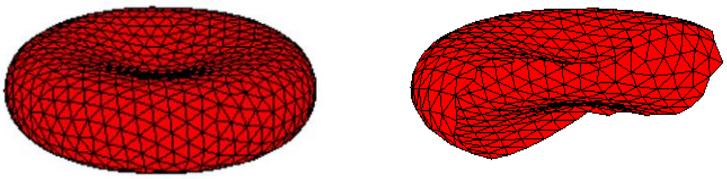


Figure 4. (1) Biconcave shape of normal human red blood cell (RBC). Modeling of RBC using mathematical expression from Kuchel (1999) in Cartesian coordinate. (2) Cross-sectional view of the 3D surface meshing model considering RBC as a thin layer of lipid bilayer sac.

To generate the initial position of the biconcave model, 3D surface meshing using implicit distance function above and Delaunay triangulation algorithm (Appendix F) is

applied. From this algorithm, we get positions of all the beads in the RBC in Cartesian coordinate and indices of three beads that forms triangle. Depending on the distance between beads that are set up, this algorithm gives different number of particles in the same overall size of RBC (see Table II).

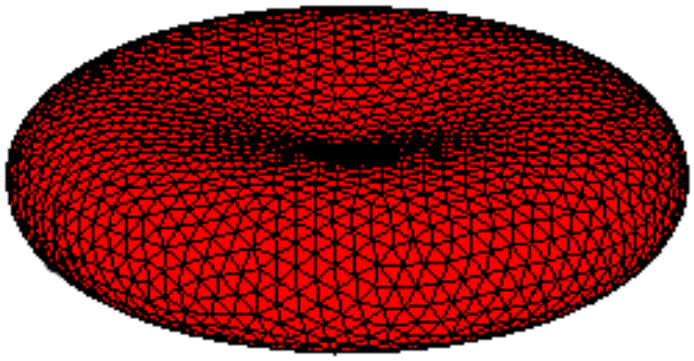
TABLE II
DISTANCE BETWEEN BEADS AND NUMBER OF PARTICLES

h_0 [a]	Number of Particles
0.4	2006
0.6	922
0.8	522
1.00	334
1.16	262
1.40	156
1.50	110

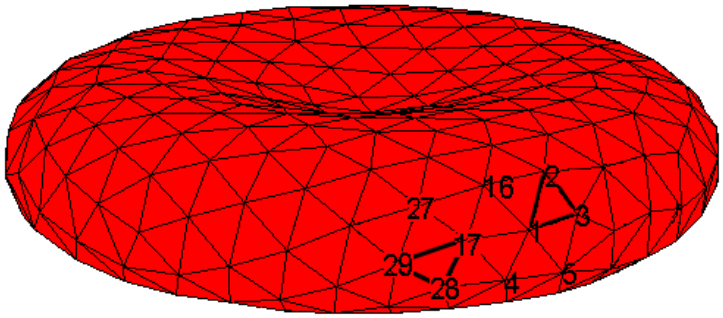
[a] distance between beads

This algorithm becomes more robust when the length between the beads are set as smaller value and/or larger bounding box so that it iterates and find the optimal curve of the function that user provides without disturbing any termination criteria within the generator. The smaller the length, the bigger number of the beads increases. For example, following shows the number of particles in RBC for $h_0 = 0.3$ and 1.16, respectively. We are able to see in Figure 5(c) that the model constructed has hexagonal lattice with pentagons and septagons.

(a)



(b)



(c)

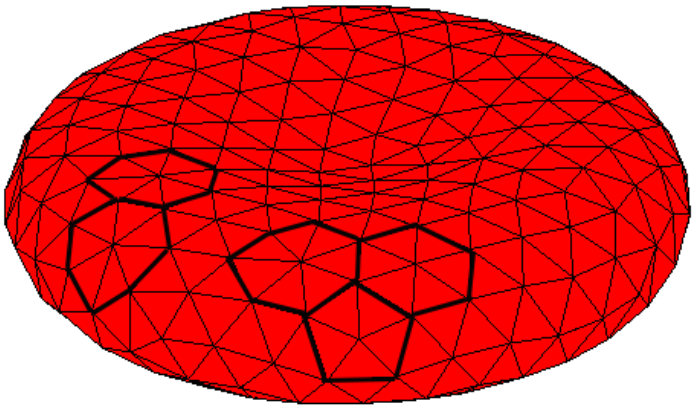


Figure 5. Example 3D surface meshing of RBC with (a) 2006 and (b) 262 hydrodynamic resistant site. (c) Hexagons, pentagons, and septagons in the lattice.

The limitation of the Persson's (2005) method is when the length h_0 is set to high value that is beyond the convergence level; In other words, that is when we try to set the model with as few beads as possible. It gets difficult to converge with fewer beads as the expression of the shape gets complicated. Nontheless, we can initiate the simulation from simple sphere with least number of beads using that method and possibly apply bending potential constraint. The bending potential would lower the energy in the center region of the cell to give biconcave shape.

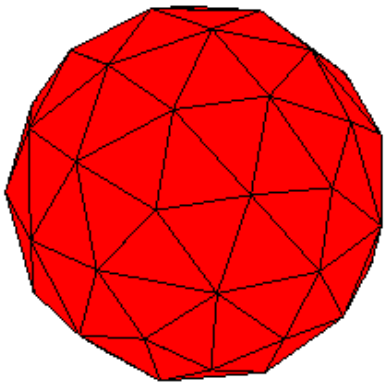


Figure 6. An example of 3D surface meshing of RBC with 78 hydrodynamic resistant sites.

Regardless of the number of beads or the shape of the model, the output workspace p contains bead positions in Cartesian coordinate in each column and t contains indices of three beads that forms triangle in each row. For the case of 262 hydrodynamic resistant sites in the RBC, it gives 520 triangles. Examples of workspaces are listed below in Table III.

TABLE III
EXAMPLE OF STORED WORKSPACES OF 262 PARTICLE CASE

(a) Workspace *t*

	A	B	C
1st triangle	1	2	3
2nd triangle	1	3	5
3rd triangle	4	1	5
4th triangle	4	5	6
5th triangle	6	5	7
6th triangle	8	6	9
7th triangle	9	6	7
8th triangle	10	8	11
9th triangle	11	8	9
10th triangle	10	11	12
...	...	...	...
517th triangle	260	238	240
518th triangle	261	242	262
519th triangle	261	240	242
520th triangle	242	241	262

(b) Workspace p

nodes	x	y	z
1	-1.54098	-3.27474	-1.35968
2	-2.51673	-3.52151	-1.25039
3	-1.75791	-4.25786	-1.14782
4	-0.54548	-3.00805	-1.33876
5	-0.7944	-3.99527	-1.31109
6	0.206042	-3.69366	-1.35591
7	-0.06192	-4.6622	-1.12149
8	1.23879	-3.33153	-1.36143
9	0.930028	-4.32063	-1.2214
10	2.393535	-3.21883	-1.32183
	...	...	...
259	-0.86469	3.304071	1.361211
260	0.118393	3.435949	1.361577
261	1.139157	3.488476	1.357507
262	2.100848	3.482507	1.312315

As in Figure 5(b) above, three indices form one triangle. Those indices that form each triangle are stored in each row in workspace t (Table III (a)). For each bead points, the position is stored in workspace p as in Table III (b). These workspaces will be read into the main simulation program.

Based on this information, total surface area can be obtained by summing all the triangles using two connector vectors for each triangle. With the list of indices, connector vectors can be defined. For inter-particle forces, unique connector vectors can be found by elimination of duplicate connectors from all the possible pair of nodes. From workspace t, we find the possible pair of nodes, sort, and eliminate duplicate.

TABLE IV
ALL POSSIBLE PAIR NODES AND UNIQUE PAIR NODES

(a)

possible pair (sorted)	
1	2
1	3
1	4
2	3
1	3
1	4
1	2
2	4
2	3
3	4
2	4
3	4
...	...

(b)

	duplicate eliminated	
k	A	B
1	1	2
2	1	3
3	1	4
4	2	3
5	2	4
6	3	4
...	...	...

Then, generalize expression for defining the unique connector vectors is

$$\boldsymbol{Q}_{\boldsymbol{unique},\boldsymbol{k}} = \boldsymbol{r}_{\mathrm{B}} - \boldsymbol{r}_{A} \tag{3.30}$$

where $\boldsymbol{r}_{\mathrm{B}}$ is position of bead with index in column B and $\boldsymbol{r}_{\mathrm{A}}$is position of bead with index in column A in the Table IV (b) above.

$$\boldsymbol{Q}_{\boldsymbol{unique},1} = \boldsymbol{r}_{2} - \boldsymbol{r}_{1} \tag{3.31a}$$

$$\boldsymbol{Q}_{\boldsymbol{unique},2} = \boldsymbol{r}_{3} - \boldsymbol{r}_{1} \tag{3.31b}$$

$$\boldsymbol{Q}_{\boldsymbol{unique},3} = \boldsymbol{r}_{4} - \boldsymbol{r}_{1} \tag{3.31c}$$

$$\boldsymbol{Q}_{\boldsymbol{unique},4} = \boldsymbol{r}_{3} - \boldsymbol{r}_{2} \tag{3.31d}$$

$$\boldsymbol{Q}_{\boldsymbol{unique},\mathbf{5}} = \boldsymbol{r}_{4} - \boldsymbol{r}_{2} \tag{3.31f}$$

$$\boldsymbol{Q}_{\boldsymbol{unique},6} = \boldsymbol{r}_{4} - \boldsymbol{r}_{3} \tag{3.31g}$$

...

Once we have tabulated data of Table IV (b), we can figure out which are the neighboring

beads for each bead position by searching for same indices. The algorithm to search for is written in the subroutine ‘InterParticleForces’ and its sub-calls in Fortran code (Appendix H). This method works for any number of beads that are greater than 3, that is, at least one or more triangles to form a RBC model. Following is an example for inter-particle forces of 4-bead model using this algorithm and definition of connector vector in Eq. (3.26).

$$F_1^{\phi} = H\left(\boldsymbol{Q}_{\boldsymbol{unique},1} + \boldsymbol{Q}_{\boldsymbol{unique},2} + \boldsymbol{Q}_{\boldsymbol{unique},3}\right) \tag{3.32a}$$

$$F_2^{\phi} = H\left(\boldsymbol{Q}_{\boldsymbol{unique},4} + \boldsymbol{Q}_{\boldsymbol{unique},5} - \boldsymbol{Q}_{\boldsymbol{unique},1}\right) \tag{3.32b}$$

$$F_3^{\phi} = H\left(\boldsymbol{Q}_{\boldsymbol{unique},6} - \boldsymbol{Q}_{\boldsymbol{unique},2} - \boldsymbol{Q}_{\boldsymbol{unique},4}\right) \tag{3.32c}$$

$$F_4^{\phi} = -H\left(\boldsymbol{Q}_{\boldsymbol{unique},3} + \boldsymbol{Q}_{\boldsymbol{unique},5} + \boldsymbol{Q}_{\boldsymbol{unique},6}\right) \tag{3.32d}$$

It proves to have same result as set of Eq. (3.27).

From the constructed RBC configuration knowing the relations of beads and its movement based on stochastic method, we are able to quantify the shear stress of the RBC flow in the solvent fluid.

3.4 Stress Tensor

It is conventional to express the total stress tensor as a sum of pressure and stress tensor where $\boldsymbol{\delta}$ is unit tensor. The extra stress tensor $\boldsymbol{\tau}$ is the part of the total stress tensor that is zero at equilibrium.

$$\boldsymbol{\pi} = p\boldsymbol{\delta} + \boldsymbol{\tau} = \boldsymbol{\pi}_s + \boldsymbol{\pi}_c \tag{3.33}$$

The total stress tensor is assumed that the contribution of serum and cells of blood are additive. Correspondingly, the isotropic pressure is $p = p_s + p_c$ and the stress tensor is

$$\boldsymbol{\tau} = \boldsymbol{\tau}_s + \boldsymbol{\tau}_c \tag{3.34}$$

The stress tensor of the solvent can be easily obtained by

$$\boldsymbol{\tau}_s = -\eta_s \dot{\boldsymbol{\gamma}} \tag{3.35}$$

where $\dot{\boldsymbol{\gamma}}$ is rate-of-strain tensor (definition in Appendix E) and $\eta_s$is the viscosity of solvent.

In general, stress tensor is a function of rate-of-strain tensor, and it can be linear or nonlinear. The constitutive equation can be expressed various ways to add viscoelaticity of the fluid. It can have differential form of rate-of-strain tensor. Examples are Jeffrey's model or retarded motion expansion which rate-of-strain tensor is expanded using Taylor series. Wedgewood (1999) developed an objective constitutive equation by expressing the stress tensor in terms of rate-of-strain tensor and the deformational vorticity tensor

$$\boldsymbol{\tau} = f(\dot{\boldsymbol{\gamma}}, \boldsymbol{\omega}_D) \tag{3.36}$$

such that there is no dependency on the reference frame. The vorticity tensor $\boldsymbol{\omega} = \nabla\boldsymbol{v} - (\nabla\boldsymbol{v})^{\dagger}$ is decomposed into the rigid vorticity tensor $\boldsymbol{\omega}_R$ part which depends on the reference frame and deformational vorticity tensor $\boldsymbol{\omega}_D$ part which is objective. As described in the introduction, Casson model is often used for blood flow calculations due to the fact that the experimental data show a slope of viscosity verses shear rate of -1/2 in the mid-region of the

shear rate where it has shear thinning effect. The model is a purely empirical model explaining the flow behavior written as

$$\sqrt{\boldsymbol{\tau}} = \sqrt{\boldsymbol{\tau}_0} + \sqrt{\eta_0}\sqrt{\dot{\boldsymbol{\gamma}}} \tag{3.37}$$

This model gives fairly good fit with the slope of exactly negative one half, however, would not explain the full range of shear rate since the model would not consider the elasticity of blood flow (Healey, 1975; Rogelio, 2007). Other Generalize Newtonian Fluid (GNF) model such as Herschel-Bulkley is used for blood flow studies. These GNF models have no elasticity. We can introduce an expression of stress tensor with elasticity based on kinetic theory.

Among all the various forms of the stress tensor, we would like to use the constitutive equation which gives physical insight into the relation between the bulk flow and the structure of polymer molecules. In this study, Kramers-Kirkwood form where the polymer contribution of the stress tensor is derived by Kramers and Kirkwood (1967) is used to find the material functions. The alternative expressions for the stress tensor appear in the (Bird *et al.*, 1987). The definitions of the material functions will be discussed in detail in the following section. Assuming that the velocity distribution is Maxwellian (see Appendix A), the polymer contribution of the stress tensor is

$$\boldsymbol{\tau}_c = -n_c \sum_v \langle \boldsymbol{R}_v \boldsymbol{F}_v^h \rangle \tag{3.38}$$

considering the Brownian motion contribution of the beads to the stress tensor $\boldsymbol{\pi}_c^b = Nn_c k_B T \boldsymbol{\delta}$ (Maxwellian); Each beads gives a contribution of $nkT\boldsymbol{\delta}$ in consequence of the equilibration in momentum space. Here, n_c is number of cells per unit volume and $\boldsymbol{R}_v = \boldsymbol{r}_v - \boldsymbol{r}_c$ is relative bead position to the center-of-mass. The bracket $\langle ... \rangle$ indicates an average with respect to the configurational distribution function over the configuration space.

When we insert the expression for the hydrodynamic force in Eq. 3.2 using Liu's (1989) stochastic approach with actual bead velocity instead of the momentum-space-averaged velocity of the bead,

$$\boldsymbol{F}_{\nu}^{h} = -\zeta \cdot [\dot{\boldsymbol{r}}_{\nu}(t) - \boldsymbol{v}_{\nu}(\boldsymbol{r}_{\nu}, t)] \tag{3.39}$$

then we obtain

$$\boldsymbol{\tau}_{c} = n\zeta\left\{\sum_{\nu}\langle \boldsymbol{R}_{\nu}\left[\dot{\boldsymbol{R}}_{\nu} - \boldsymbol{\kappa}\cdot\boldsymbol{R}_{\nu}\right]\rangle + \langle \boldsymbol{R}_{\nu}[\dot{\boldsymbol{r}}_{c} - \boldsymbol{\kappa}\cdot\boldsymbol{r}_{c}]\rangle\right\} \tag{3.40}$$

Only the first term of Eq. 3.37 survives if we neglect the hydrodynamic interaction. This is due to the center-of-mass moving on the average with the fluid velocity. The stress tensor $\boldsymbol{\tau}_{c}$ is symmetric with no external force being present; therefore, we can rewrite the equation as

$$\boldsymbol{\tau}_{c} = \frac{1}{2}n\zeta\sum_{\nu}\left\{\langle \boldsymbol{R}_{\nu}\left[\dot{\boldsymbol{R}}_{\nu} - \boldsymbol{\kappa}\cdot\boldsymbol{R}_{\nu}\right]\rangle + \langle\left[\dot{\boldsymbol{R}}_{\nu} - \boldsymbol{\kappa}\cdot\boldsymbol{R}_{\nu}\right]\boldsymbol{R}_{\nu}\rangle\right\} \tag{3.41}$$

Then using the definition of convected derivative

$$\boldsymbol{\Lambda}_{(1)} = \frac{d}{dt}\boldsymbol{\Lambda} - \{\boldsymbol{\kappa}\cdot\boldsymbol{\Lambda} + \boldsymbol{\Lambda}\cdot\boldsymbol{\kappa}^{\dagger}\}$$

we get

$$\begin{aligned}\boldsymbol{\tau}_{c} &= \frac{1}{2}n\zeta\sum_{\nu}\left\{\frac{d}{dt}\langle \boldsymbol{R}_{\nu}\boldsymbol{R}_{\nu}\rangle - \boldsymbol{\kappa}\cdot\langle \boldsymbol{R}_{\nu}\boldsymbol{R}_{\nu}\rangle - \langle \boldsymbol{R}_{\nu}\boldsymbol{R}_{\nu}\rangle\cdot\boldsymbol{\kappa}^{\dagger}\right\} \\ &= \frac{1}{2}n\zeta\sum_{\nu}\langle \boldsymbol{R}_{\nu}\boldsymbol{R}_{\nu}\rangle_{(1)}\end{aligned} \tag{3.42}$$

For steady state, the polymer contribution of stress tensor is

$$\boldsymbol{\tau}_{c} = -\frac{1}{2}n\zeta\sum_{\nu}\{\boldsymbol{\kappa}\cdot\langle \boldsymbol{R}_{\nu}\boldsymbol{R}_{\nu}\rangle + \langle \boldsymbol{R}_{\nu}\boldsymbol{R}_{\nu}\rangle\cdot\boldsymbol{\kappa}^{\dagger}\} \tag{3.43}$$

The Eqs. 3.42 and 3.43 are a stress tensor derived by Kramers-Kirkwood form that applies to any flow types.

3.5 Material Properties

The rheological material functions relate the kinematics of a flow to the stresses required to sustain the motion. The shear flow and shear free elongational flow are often used to characterize polymeric liquids. We will focus on these two types of flow conditions to obtain the material properties of RBC. All flows are considered homogeneous and incompressible (see Appendix E) so that the deformations are uniform. The Cartesian coordinates are chosen for all the calculations that are done in this paper.

3.5.1 Shear Flow

When a flow is in x direction, the gradient direction in y, and the vorticity in z, we can postulate the velocity field as follow:

$$v_x = \dot{\gamma}_{yx}(t)y, \qquad v_y = 0, \qquad v_z = 0 \tag{3.44}$$

in which the rate-of-strain $\dot{\gamma}_{yx}$ can be a function of time and the absolute value of $\dot{\gamma}_{yx}$ is the shear rate $\dot{\gamma}$. The viscosity, η, which measures shear stress is a function of shear rate as oppose to a constant in Newtonian flow. In the non-Newtonian flow, we introduce additional material functions such as the first normal stress coefficient and the second normal stress coefficient, ψ_1 and ψ_2 respectively. Particularly in the steady shear flow, the second normal stress ψ_2 becomes zero.

$$\tau_{yx} = -\eta(t,\dot{\gamma})\dot{\gamma}_{yx} \tag{3.45a}$$

$$\tau_{xx} - \tau_{yy} = -\psi_1(t,\dot{\gamma})\dot{\gamma}_{yx}^2 \tag{3.45b}$$

$$\tau_{yy} - \tau_{zz} = -\psi_2(t,\dot{\gamma})\dot{\gamma}_{yx}^2 \tag{3.45c}$$

These are differences of normal stress tensors that can be measured since the τ_{ii} components are not zero (Eq. 3.46). The total stress tensor in the Eq. 3.33 is

$$\boldsymbol{\pi} = \begin{pmatrix} p + \tau_{xx} & \tau_{yx} & 0 \\ \tau_{yx} & p + \tau_{yy} & 0 \\ 0 & 0 & p + \tau_{zz} \end{pmatrix} \tag{3.46}$$

for a simple shear flow. Each component of stress tensor and invariants of $\dot{\boldsymbol{\gamma}}$ are shown in Appendix E.

For time-dependent unsteady start-up flow, the stress tensor with the transient term in Eq. 3.42 is used. The material functions for such flow are noted as $\eta^{+}(t,\dot{\gamma})$, ${\psi_1}^{+}(t,\dot{\gamma})$, and ${\psi_2}^{+}(t,\dot{\gamma})$ where $\dot{\gamma} = 0$ at $t < 0$. For the steady state flow, time-dependent term in Eq. 3.42 vanishes to Eq. 3.43. Then, the functions are $\eta(\dot{\gamma})$, $\psi_1(\dot{\gamma})$, and $\psi_2(\dot{\gamma})$.

3.5.2 Stress Relaxation after Sudden Shearing Displacement

The fluid is at rest for all times previous to $t = 0$.

$$u_x = 0, \qquad u_y = 0, \qquad u_z = 0 \tag{3.47a}$$

We can postulate the sudden displacement as follow:

$$u_x = \gamma_0 y, \qquad u_y = 0, \qquad u_z = 0 \tag{3.47b}$$

where γ_0 is shear strain that can be induced by applying a large, constant shear rate $\dot{\gamma}_0$ for a short time interval Δt so that $\dot{\gamma}_0 \Delta t = \gamma_0$. The time decay of the shear stress is described by the relaxation modulus $G(t,\gamma_0)$ and the relaxation of the first normal stress difference by the function $G_{\psi_1}(t,\gamma_0)$ (Eq. 3.48a and Eq. 3.48b)

$$\tau_{yx} = -G(t,\gamma_0)\gamma_0 \tag{3.48a}$$

$$\tau_{xx} - \tau_{yy} = -G_{\psi_1}(t,\gamma_0){\gamma_0}^2 \tag{3.48b}$$

For a small shear strain limit, the relaxation modulus becomes independent of γ_0 and contains the same linear viscoelastic information as elastic modulus G' and G''.

3.5.3 Shear Free (Elongational) Flow

The shear free flow shows lamina stretching rather than shear. The velocity field of elongational flow is expressed as follow with the elongation rate $\dot{\varepsilon}$ which can depend on time.

$$v_x = -\frac{1}{2}\dot{\varepsilon}x, \qquad v_y = -\frac{1}{2}\dot{\varepsilon}y, \quad v_z = +\dot{\varepsilon}z \tag{3.49}$$

There are two independent combinations of stress tensor component that are in interest for incompressible fluids since the total stress tensor in the Eq. 3.33 has only the diagonal component. This means that the deformation is unaffected by the rotation about the axes.

$$\tau_{zz} - \tau_{xx} = -\bar{\eta}_1(\dot{\varepsilon})\dot{\varepsilon} \tag{3.50a}$$

$$\tau_{yy} - \tau_{xx} = -\bar{\eta}_2(\dot{\varepsilon})\dot{\varepsilon} \tag{3.50b}$$

For the elongational and biaxial stretching flows as introduced in the Appendix E, the x and y directions are indistinguishable.

$$\bar{\eta} = \bar{\eta}_1 \tag{3.51a}$$

$$\bar{\eta}_2 = 0 \tag{3.51b}$$

Hence, we only calculate one viscosity function $\bar{\eta}_1$ in the simulation for shear free flow.

3.6 Radius of Gyration

The gyration of radius tells how much the polymers are stretched out from center of mass. The mean-square radius of gyration is

$$\langle s^2 \rangle = \langle \frac{1}{N}\sum_{\nu}^{N}(\boldsymbol{R}_\nu \cdot \boldsymbol{R}_\nu) \rangle \tag{3.52}$$

$\langle s^2 \rangle_{eq}$ is the mean-square radius of gyration at equilibrium. In other words, it is no-flow with infinite wall condition.

We now see how these rheological material functions are calculated in the simulation code in detail.

4. SIMULATION

Based on the theory, the simulation code is developed. The realization of a trajectory requires the sampling of random numbers. Uniform random numbers are used rather than Gaussian distributed since the computational cost is significant for BD simulations.

The Figure 7 shows a flow chart of the program. Sample Fortran codes are given in Appendix H.

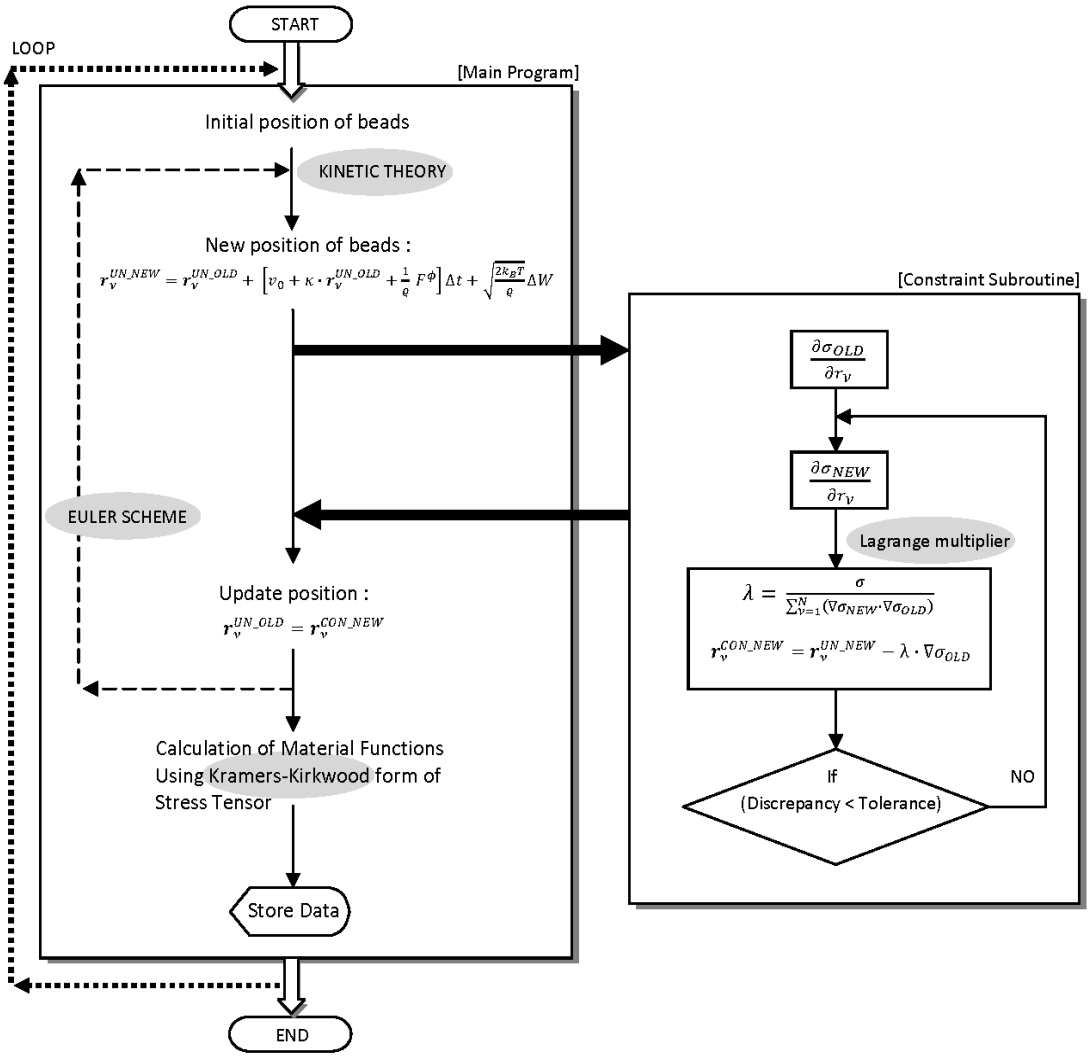


Figure 7. Flow chart of the program.

The two constraints method was tested with the triangular model. To initiate the configuration of the model, the simulation is run until the system reaches the steady state in flow. For each time step, five values of viscosity that are collected previously are used to determine the plateau point of steady state with the criteria of $\mathrm{tol}_{\mathrm{slope}}$. The material functions can then be obtained for steady state from the bead positions. This process is repeated for each shear rate. The data for steady state shown in the result section is extrapolated to $\Delta t = 0$. Linear least square method is employed for the determination of steady state and for extrapolation.

TABLE V
THE PARAMETERS USED FOR THE SIMULATIONS

Parameter	Value	Description
$\dot{\gamma}_{yx}$	10^{-2}~10^{3}	dimensionless shear rate for shear flow
$\dot{\varepsilon}$	10^{-3}~ 4×10^{1}	dimensionless elongational rate for shear free flow
$\mathrm{N}_{\mathrm{traj}}$	$10^{4} \sim 10^{6}$~	number of trajectories
Δt	10^{-4}~10^{-6}	dimensionless time step size
$\mathrm{tol}_{\sigma_1,\sigma_2}$	1.5×10^{-3}	convergence criteria for constraints
a, a', a''	$100 \sim 400$	constant area of $\sum(\mathrm{area})^2$
l		Constant length of $\sum(\boldsymbol{Q}_{\mathrm{i}})^2$ when initial area = a
b		constant volume when initial area = a'

The initial parameter settings are given above in Table I. Initial position is set up so that the area is equal to a constant a given in Table V. This value is determined by the relation between the size of the area of triangle and the slope of the viscosity to the shear rate. To optimize value of the area of triangle, different sizes of triangle were tested by Qin 2008. The slope is compared with the empirical data result (Dintanfess, 1974). The perimeter-constraint is sum of length square which is $|\boldsymbol{Q}_1|^2 + |\boldsymbol{Q}_2|^2 + |\boldsymbol{Q}_3|^2$ as described in Section 3.3.1. The value of this parameter is initially set up from the initial position. If the initial position is

equilateral triangle with $(\text{area})^2 = a$ of 100, the sum of length square (l) is 69.2820323. Since this condition gives the shortest length, the model would be too rigid. Therefore, in this study the value in Table V is used.

For triangle model with one constraint, the range of the dimensionless time step Δt was set between 10^{-2}~10^{-5} depending on the shear rate. The $\Delta t = 10^{-2}$ would be small enough for low shear rates. However, as more beads and springs are added to the configuration, 10^{-2} would give a larger error and would not converge. With the trial run, Δt needs to be set to 10^{-4} for triangle with two constraints. As the shear rate increases, smaller Δt is needed requiring more integration steps. Higher shear rate converges slower and requires dimensionless time step Δt value closer to zero to converge, and vice versa. Larger Δt decrease the calculation time, but leads to larger deviations and possibly non-convergence. Despite of slower convergence of higher shear rate, it is calculated first to reduce the overall simulation time because at high shear rate the model reaches the steady state faster in laminar flow. Then the simulation runs for the next shear rate starting from the time where the previous shear rate reached the steady state. This process reduces greatly the number of time integration steps and unnecessary constraint iteration so that we can find the steady state reaching point faster in the simulation.

All calculations were executed on the Extreme Computing System with Intel® Xeon® CPU E5-2670 0 at 2.60GHz processor and ARGO Beowulf cluster with Dual AMD Opteron, two 3GHz Xeon, and two dual-core AMD Opteron processors.

The results on the rheological properties of the model with two constraints under different flow conditions are discussed in the following section.

5. RESULTS and DISCUSSION

5.1 Three-Bead-Spring Ring Model with One Constraint

In this section, the results of Lopez's one-constraint model with optimized value of the area of triangle are shown. The transient term and no-flow condition (in otherwords, no shear) in the beginning of the flow is added in the simulation code. The start-up shear flow is observed for longer dimensionless time ($t = 0.5{\sim}50$) to ensure that the steady-state is reached. The steady state result is a snapshot result at $t = 50$.

5.1.1 Stress Relaxation

In the stress relaxation experiment, we observe relaxing stress after sudden shearing displacement is applied. A fluid sample with the three beads-springs ring under constant area constraint that is in no flow condition is suddenly applied with shear strain γ_0. This can be induced by a large, constant shear rate $\dot{\gamma}_0$ for a short time interval. The time interval used in the simulation is $\Delta t = 0.0002$. The response observed is that the model is back to its equilibrium size in a short, but finite, time (Thurston, 2004).

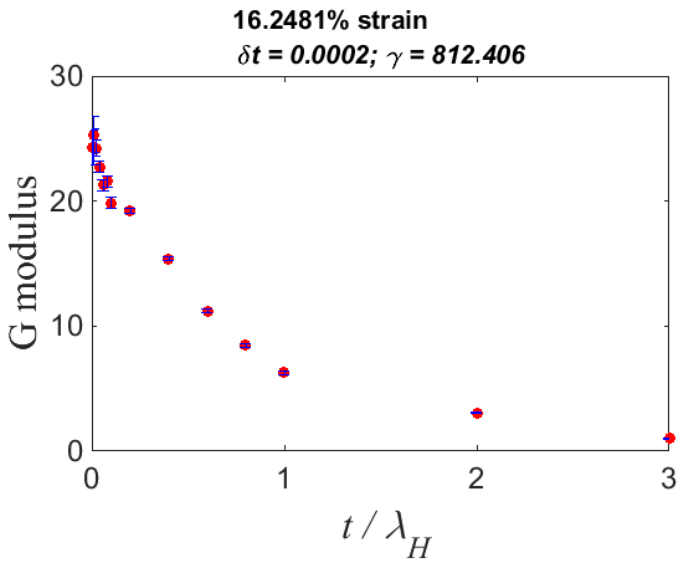


Figure 8. Relaxation of elastic modulus with 16.25% strain ($a = 100$).

The RBC cell model recovers its shape to equilibrium condition such that the slope of log plot of relaxation modulus verses dimensionless time ranges from -3.4268 to -1.0686 which gives the relaxation time measured ranging 0.2829 to 0.9358 λ_H. This time scale is used to compare with the experiment data for this particular model. Because it is a displacement experiment, there is no significant change in the radius of gyration.

5.1.2 Start-up Shear Flow

The shear rate range between 0.1 and 50 were tested. For higher shear rate, it required smaller time steps. In other words, it took more looping in the simulation for integration resulting longer physical time to simulate. However, as shown in Figure 9, it is proven that it takes less dimensionless time for the higher shear rate to reach the steady state than the lower shear rate when we plot the result. The unit of viscosity becomes dimensionless when divided by the

number of cells, the k_BT factor, and time constant for Hookean spring. The unit of time divided by time constant makes it dimensionless.

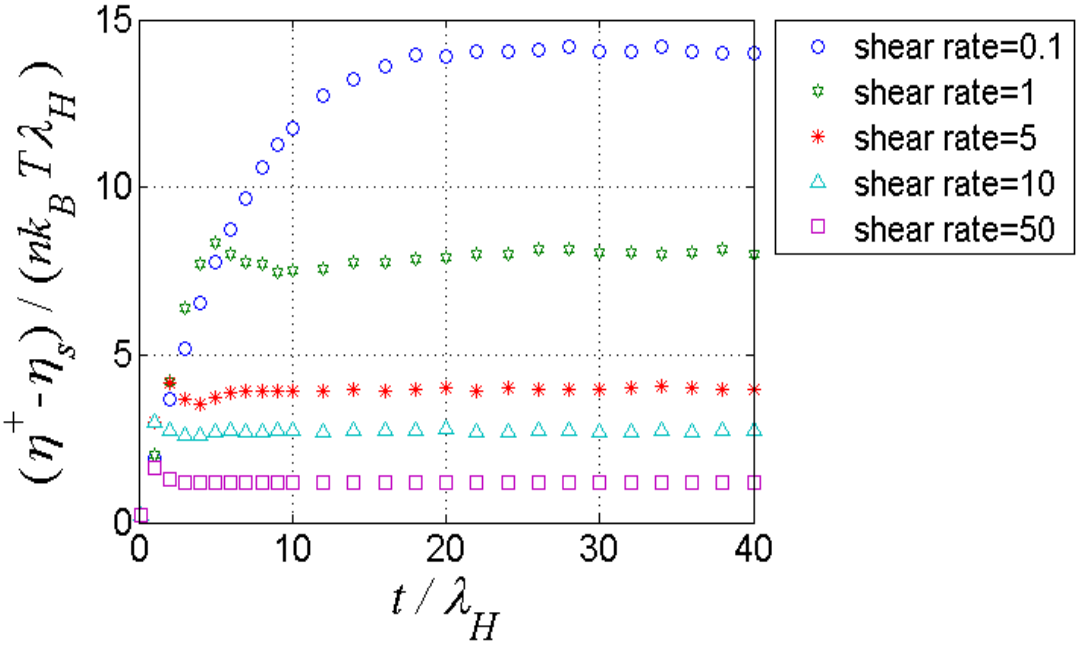


Figure 9. Viscosity of a suspension of bead-spring triangle with one-constraint plotted verses dimensionless time for a start-up shear flow ($a = 100$).

In the higher shear rate, overshoot is shown before it approaches steady value. The size of the maximum overshoot is larger with increasing shear rate. In general, the viscosity decreases as shear rate increases, and further observation will be discussed in the steady state shear flow section.

5.1.3 Steady Shear Flow

As shown in Bird *et al.* (1987), neither of the Rouse nor the Zimm model result gives a decrease of viscosity with increasing shear rate. Liu (1989) demonstrated the result of Kramers chain, a bead-rod model, with multiple constraints showing the shear thinning effect. The bead-spring-ring model with an area constraint was constructed to represent the RBC microstructures in this preliminary study and the results clearly show the expected shear-

thinning effect for viscosity. Comparison of viscosity with experimental data (Copley, 1973; Dintanfass, 1974) is shown in Fig. 10.

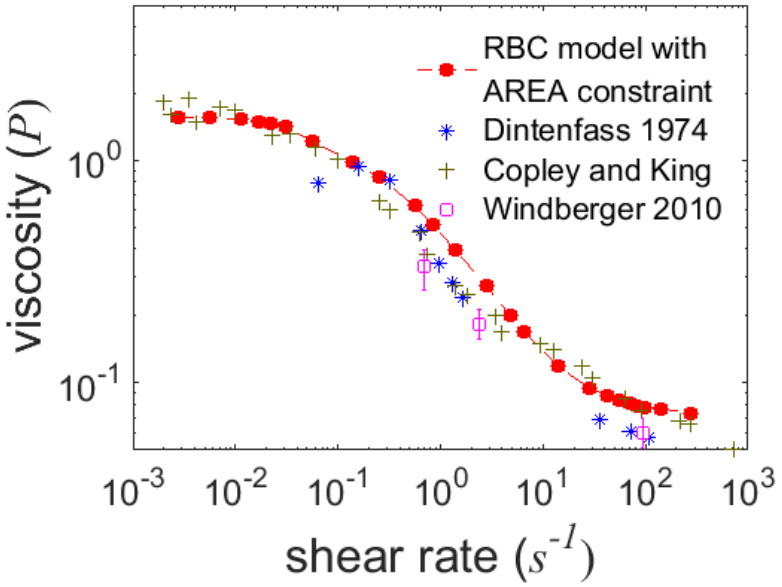


Figure 10. Comparison of the one-constraint triangular model result against the experimental data. (a) The simulated results of a simple bead-spring-ring model with one constraint ($a = 100$ dimensionless unit). (b) Dintanfass (1971, 1974) used cone-in-cone rhombospheroid viscometer without the use of anticoagulant. The samples were tested immediately after withdrawal. (c) Copley and King (1973) used Weissenburg rheogoniometer with the use of dry ethylenediamine tetraacetate (EDTA) as an anticoagulant. The blood samples were that of human donors ranging in age from 25 to 60 years. (d) Windberger (2010) used Cell-Dyn 3500 and K-EDTA after the withdrawal.

The scaled simulation result of bead-spring-ring model above has two adjusted parameters: the low-shear-rate plateau value and the constraint area of the model. The time scale value λ is from the relaxation simulation. This result shows that this simplest bead-spring-ring configuration gives the quantitatively similar trend compare to the experimental observation by Dintanfess (1974) and Copley (1973), thus gives the potential of BD with constraint algorithm for future studies of blood flow. In the mid region between 0.1 $\sec^{-1}$and 50 $\sec^{-1}$, the viscosity depends on the rate of shear showing non-Newtonian shear thinning where the

viscosity decreases as shear rate increase. The slope is approximately negative one half in the log-log plot of shear rate verses viscosity.

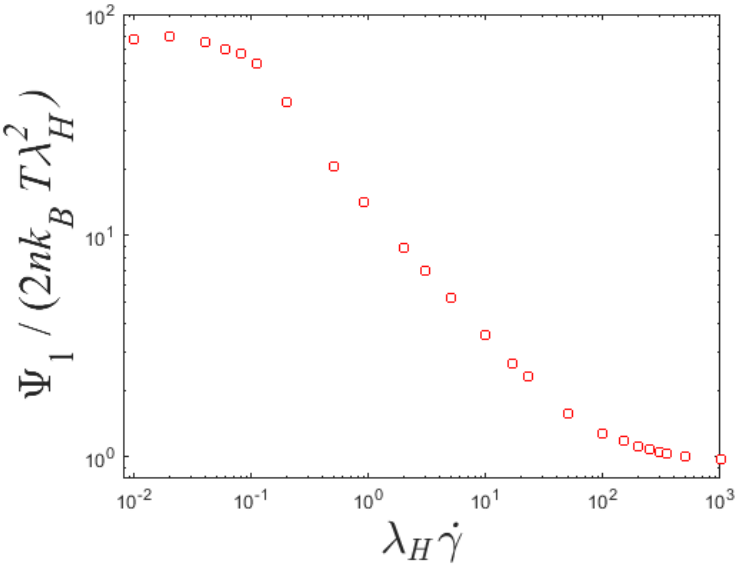


Figure 11. A log-log plot of the first normal stress of a suspension of bead-spring triangle with one-constraint with respect to shear rate for a steady shear flow ($a = 100$).

In the region after 50 $\sec^{-1}$, all the material functions η and ψ_1 approach plateau values showing the behavior of Newtonian fluid. The second normal stress coefficient $\psi_2$is zero in all range of shear rate. The plot of first normal stress coefficient is shown in Figure 11.

Different sizes of area are compared for optimization. The following plot in Figure 12 is the viscosity and first normal stress difference of different area size for single area constraint. When we increase the size of the area, the absolute value of the slope increased as the yx-component of shear stress increases.

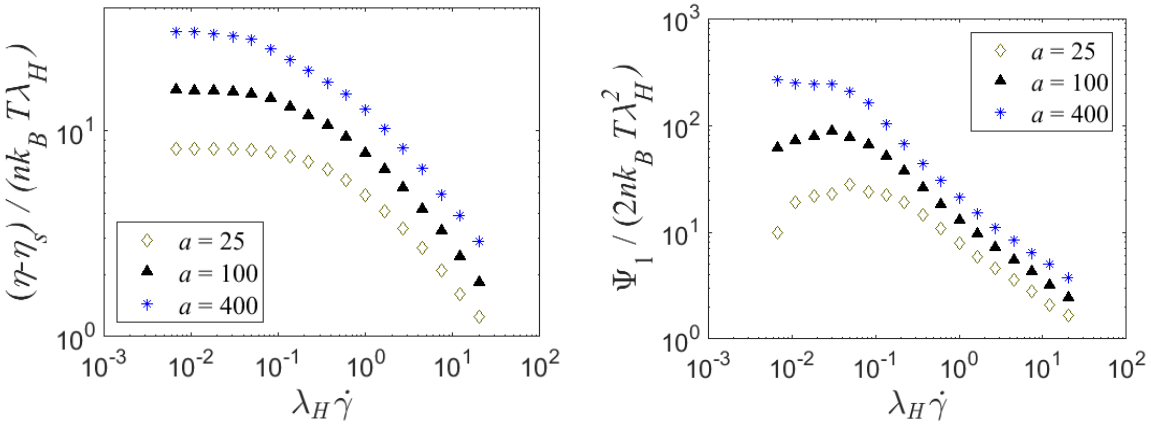


Figure 12. A comparison of result of one-constraint method with different area sizes. (a) A log-log plot of viscosity with respect to shear rate for a steady shear flow. (b) A log-log plot of first normal stress coefficient with respect to shear rate for a steady shear flow.

The value of the slope for each size of area is compared in Table VII. We have chosen the area that matches the experimental data. That value is used throughout this study and is one of the two adjusted parameters that can be reoptimized for multi-bead model in the future study. Generally speaking, the size of erythrocyte depends on mammalian species (Windberger, 2010). For the method comparison purpose, we do not variate the size of the area on the rest of the study.

5.1.4 Capillary Flow

In microcirculation, the diameters of the vessels smaller than $100\ \mu m$ is considered as a capillary flow. The position of the beads in boundary flow can be calculated with either reflection or rejection method (Fig. 13). The reflection method is done by repositioning the position of the bead that is out of boundary and locates it within the boundary by reflecting the position with remaining beads as an axis. This shows that the microstructure bounces on the wall. The rejection method rejects the positions of the microstructure that are out of

boundary and recalculates the positions of the beads so that all the beads are within the boundary.

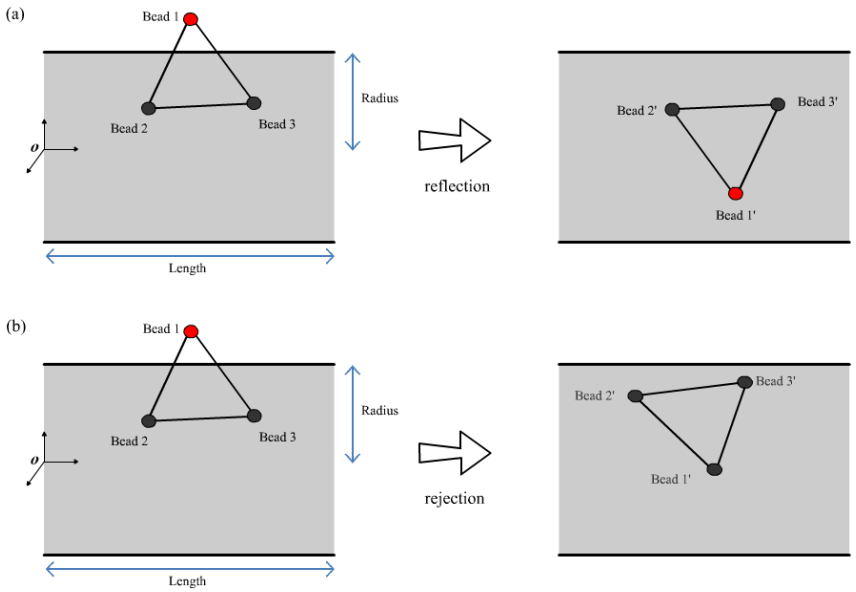


Figure 13. (a) Representation of reflection method. (b) Rejection method for capillary flow.

In the micro vessels, the RBCs tend to migrate toward the center of the tube. This is due to tendency of the RBCs to move faster toward the center of the micro vessel while move slower near the wall. As the velocity of flow increases, the volume percentage of RBCs (hematocrit) increases in the center (Fahraeus, 1931). The simulation methods for capillary flow described above comply with this fact because these methods increase the chance of RBCs to be located towards the center when the calculation of the position is out of boundary.

In this study, the rejection method is used for several reasons. We do not want to consider elastic vessel wall to be solid and the RBC particle as well. Moreover, we are not absolutely sure how the pulse affects the position of the cells. The result below (Fig. 14) is capillary flow using rejection method with the dimensionless radius of the vessel ranging

1~40 for given shear rate of 0.5. It shows that the viscosity drops drastically as the dimension decreases.

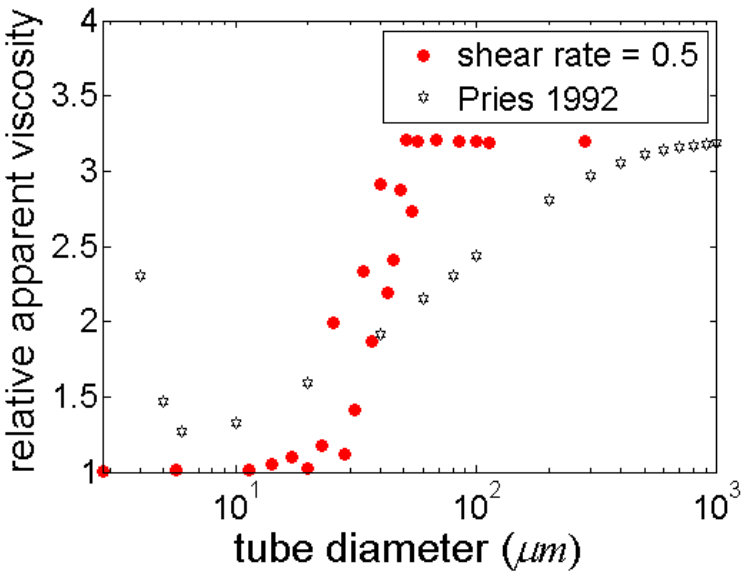


Figure 14. Fahraeus-Lindqvist effect is observed in a capillary flow. Compared with Pries (1992) analysis ($a = 100$).

We have observed that the blood has shear thinning effect. It can also be considered as thixotropic which the viscosity not only depends on the shear rate, but also on the time of shearing. This can explain the behavior in the micro vessels where the fluid gets less viscous as it slides through the capillary because the fluid is agitated or stressed. This is commonly known as Fahraeus-Lindqvist effect (Fahraeus, 1931). We can also verify this effect by observing the position of the RBC migrating to the center of the domain in the radial direction. The relative viscosity of the bead-spring triangle model decreases when the radius of the vessel decreases in steady shear flow as shown in the analysis of experimental data by Pries (1992). Pries has shown vessel diameter dependence on the relative apparent viscosity analyzing the experimental measurements mostly tube hematocrit, the volume concentration of RBC within the tube, between 0.4 and 0.45. It is stated that the influence of shear rate on

viscosity appears to be small in the microcirculation under normal conditions generally above 50 $\sec^{-1}$. At substantially lower shear rate, it shows significant effects on viscosity which the reasons can be that the tube flow is stongly affected by cell segmentation and effect of RBC aggregation tendency. Therefore, it is advised that we simulate shear rate above 50 $\sec^{-1}$ to give fair comparison.

5.1.5 Steady Shear Free Flow

The plot of elongational viscosity $\bar{\eta}$ verses elongational rate is shown in Figure 15.

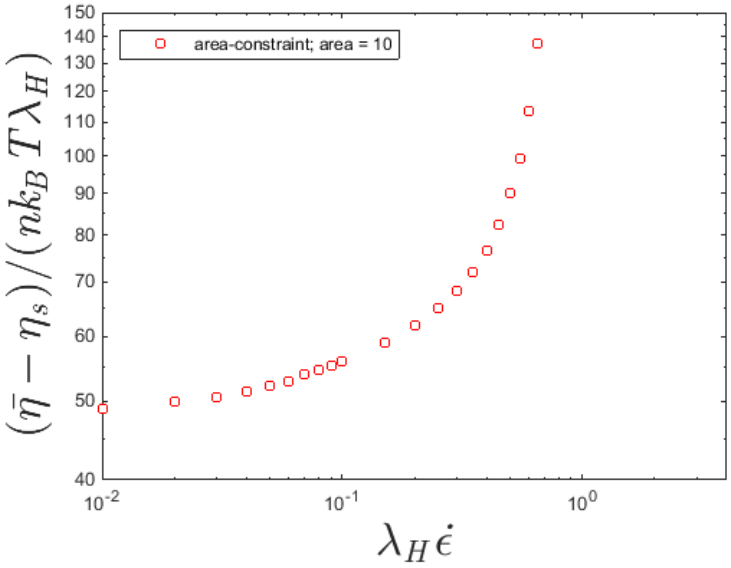


Figure 15. A log-log plot of viscosity of a suspension of bead-spring triangle with one-constraint with respect to the elongation rate for a shear free flow ($a = 100$).

It is seen that the viscosity increases drastically as the elongational rate increases in the log-log plot. At low elongational rates the viscosity approaches a constant value $\bar{\eta}_0$ known as the zero-elongation-rate viscosity. This value is known to have three times the zero-shear-rate viscosity.

TABLE VI
COMPARISON OF ZERO-RATE VISCOSITY

(a) (b)

SHEAR FLOW		SHEAR-FREE FLOW	
Shear rate	Viscosity	Elongational rate	Viscosity
0.01	15.5671	0.01	48.93449
0.02	15.5642	0.02	49.83773
0.04	15.4459	0.03	50.58737
0.06	14.8614	0.04	51.38912
0.08	14.6285	0.05	52.18082
0.11	14.2861	0.06	52.77896
		0.07	53.91829
		0.08	54.62412
		0.09	55.18893
		0.10	55.90995
Zero-shear-rate	15.848	Zero-elongation-rate	48.248

When the viscosity of shear and shear-free data between $0.01 \sim 0.11$ is extrapolated (Table VI), the value of extrapolated zero-shear-rate times three is 47.544 which is approximately the value of extrapolated zero-elongation-rate.

5.2 Three-Bead-Spring Ring Model with Two Constraints

The two-constraint model of three-bead-spring ring is run for dimensionless time range of $0.5 \sim 20$ for a start-up flow. The steady state result is collected at the time where the steady state is reached. The result of steady state flow with two-constraint method is compared with that of one-constraint. To optimize value of the area of triangle, three different sizes of triangle are tested.

5.2.1 Stress Relaxation

For the relaxation simulation of combined constraint model, it was difficult to observe decay of relaxation modulus value under 20% strain due to non-linearity in the constraint. The RBC cell model recovers its shape to equilibrium condition such that the slope of log plot of relaxation modulus verses dimensionless time ranges from -92.717 to - 3487.9 which gives the relaxation time measured ranging approximately 0.000287 to 0.01 λ_H (Fig. 16).

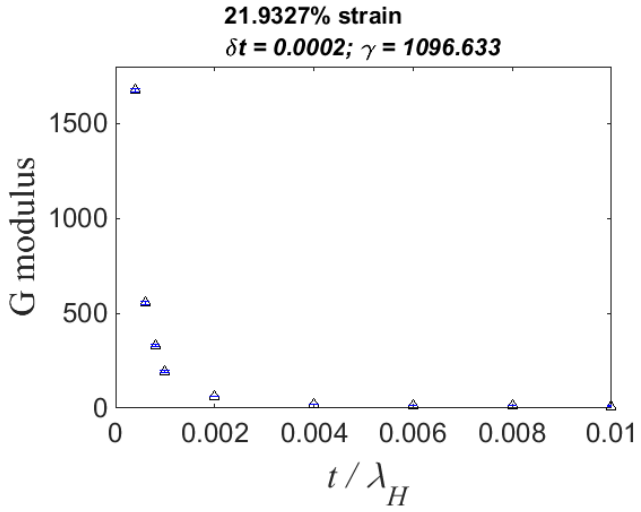


Figure 16. Relaxation of elastic modulus with 21.93% strain ($a = 100$).

This shows that the model recovered much faster than the area only constraint which means the constraints affected more to this model.

5.2.2 Start-up Flow

We have seen in Figure 9 that the higher shear rate reaches steady-state faster than lower shear rate. Instead of calculating all the points for time range of $0.5\sim40$ and collect steady state points at $t = 40$ for all shear rates, the calculation of highest shear rate is calculated first

starting at $t = 0.5$. After letting it flow for a while, five data points are collected to check if the flow reached the steady state by linear least square method. When it reached steady state, the next shear rate calculation starts from the time point where the previous shear rate reached steady state.

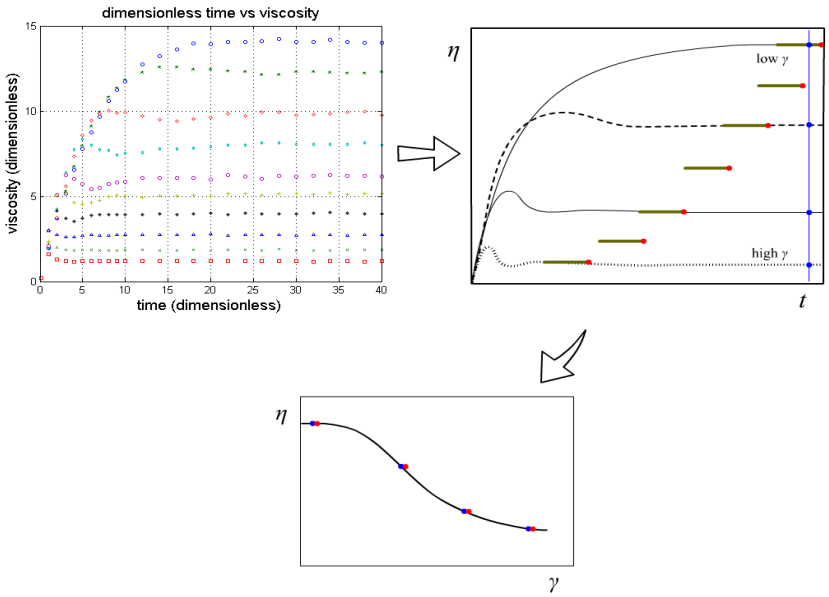


Figure 17. Representation of how data are collected for two-constraint method to reduce computer running time.

Viscosity of a suspension of bead-spring triangle with two-constraint model plotted verses dimensionless time for a start-up shear flow.

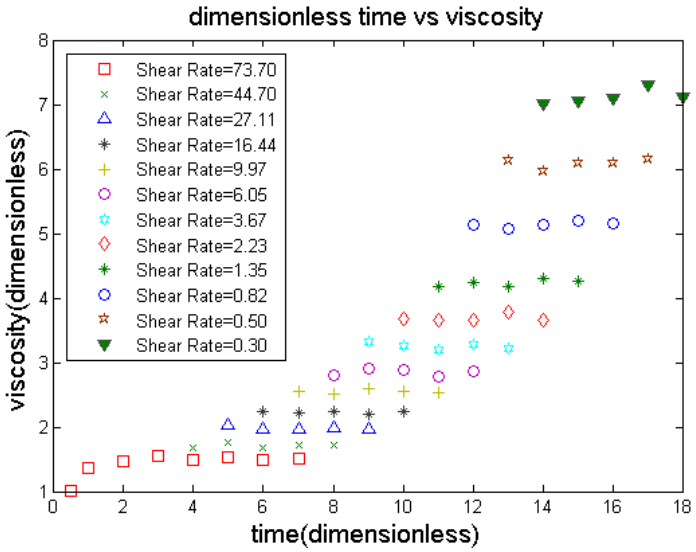


Figure 18. Viscosity of a suspension of bead-spring triangle with two-constraint model plotted verses dimensionless time for a start-up shear flow ($a = 100$).

The Figure 17 shows how the data is collected and the reason why the result of two-constraint method in Figure 18 appears different from Figure 9. In fact, the trend of two-constraint method should be similar to that of one-constraint. By using this method, the overall calculation time is drastically reduced.

5.2.3 Steady Shear Flow

The Figure 19 is the result of two constraint method with triangular model in shear flow collecting the steady state data from the last points of start-up flow. The sample selected data are in the table in Appendix G.

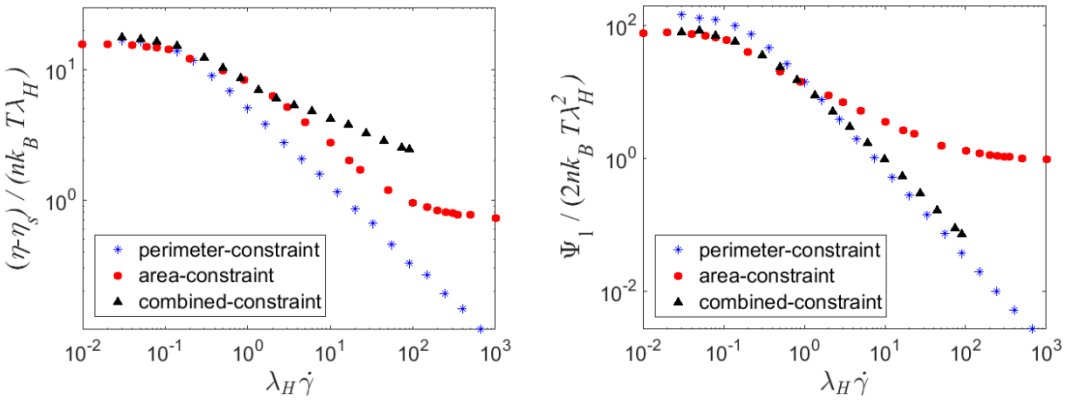


Figure 19. A comparison of result of two-constraint method against one-constraint of a triangle model ($a = 100$). (a) A log-log plot of viscosity with respect to shear rate for a steady shear flow. (b) A log-log plot of first normal stress coefficient with respect to shear rate for a steady shear flow.

The result of two-constraint was superposed to that of one-constraint for comparison. In Table VII, the slope of each case is presented. Only a few data points in the mid-section of shear rate verses viscosity where the viscosity value drops significantly is selected to calculate the slope. The result with area and sum of length square constraint in Figure 19 shows that the slope drops to -0.2718 in the mid region between dimensionless shear rate $1\sim10$ showing less shear thinning. In addition, the high shear rate limit did not show clear plateau. This is due to the elimination of degree of freedom that the model does not have flexibility to change its shape compare to one constraint. Our interest for the preliminary model was to see whether the two-constraint method converges well. In addition, in the case of perimeter-constraint only, the model is insufficient to get result for higher shear rate in the simple shear flow. We have observed that the model just fold with no area.

TABLE VII
OPTIMIZATION OF AREA SIZE

	EXPERIMENTAL DATA		ONE-CONSTRAINT			TWO-CONSTRAINT		
	Dintenfass [a]	Copley and King [b]	Size of area [c]			Size of area [d]		
			25	100	400	25	100	225
Slope	-0.5106	-0.5188~-0.5212	-0.4214	-0.5217	-0.6151	-0.2458	-0.2718	-0.2793

[a] Dintenfass, L.: *Biorheology*, 11: 397, 1974.
[b] Copley, A.L.: *Biorheology*, 10: 87, 1973.
[c d] size of area is dimensionless and the value is $\sum(\text{area})^2$

In the proposed tetrahedron model with volume and total area constraint, there would we more degrees of freedom than the triangular model with area and sum of length square constraint since there are four areas of triangle added to give constant area. The size of the area is expected to be optimized to fit the experimental data.

Viscosity and first normal stress coefficient of different sizes of area are compared for the combined constraint.

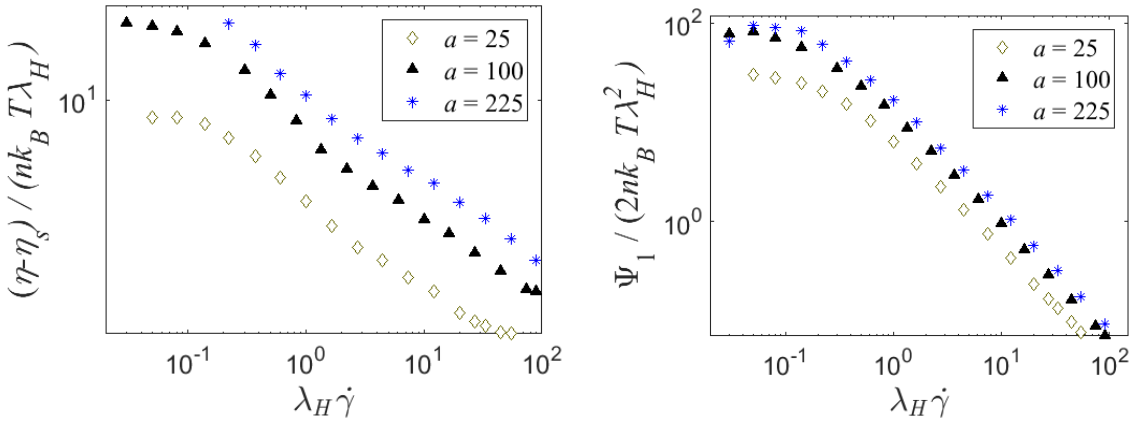


Figure 20. A comparison of result of two-constraint method with different area sizes. (a) A log-log plot of viscosity with respect to shear rate for a steady shear flow. (b) A log-log plot of first normal stress coefficient with respect to shear rate for a steady shear flow.

To obtain the value in the high shear rate beyond dimensionless unit 150 in Figure 20, smaller Δt is needed to retain the constraint; otherwise, the calculation result gives non-convergence. Table VII also shows that there are no significant differences in slope against different sizes.

5.2.4 Steady Shear Free Flow

The results for the RBC model under constraint for steady elongational flow are presented in Figure 21. This graph shows the comparison between single constrained method and combined constraint method for elongational flow with same area size.

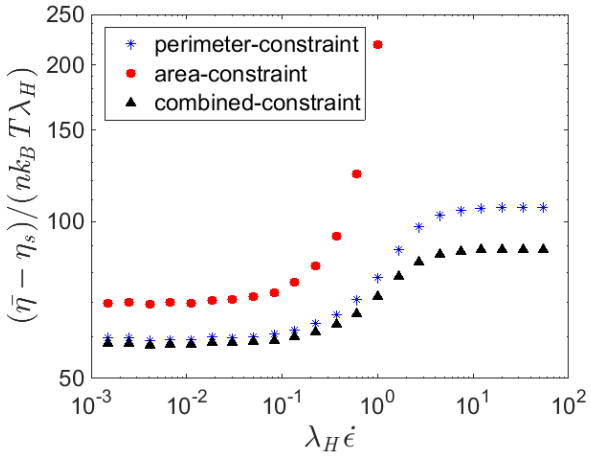


Figure 21. A comparison log-log plot of elongational viscosity $\bar{\eta}$ as a function of the elongation rate $\dot{\epsilon}$ for the triangular model with constraints ($a = 100$).

At low elongational rate $\dot{\epsilon}$, the elongational viscosity $\bar{\eta}$ approaches a constant value $\bar{\eta}_0$ known as the zero-elongation-rate viscosity. This value is known to have three times the zero-shear-rate viscosity. When the viscosity of shear and shear-free data are extrapolated to zero-rate point respectively, the value of extrapolated zero-elongation-rate approximately matches when the value of extrapolated zero-shear-rate is multiplied by three. As the elongational rate is increased, $\bar{\eta}(\dot{\varepsilon})$ is seen to increase. There is no data available of typical

behavior for blood. At higher elongational rate, our model with single constraint is unable to maintain its area integrity; thus we are unable to measure the elongational viscosity plateau value that we would expect.

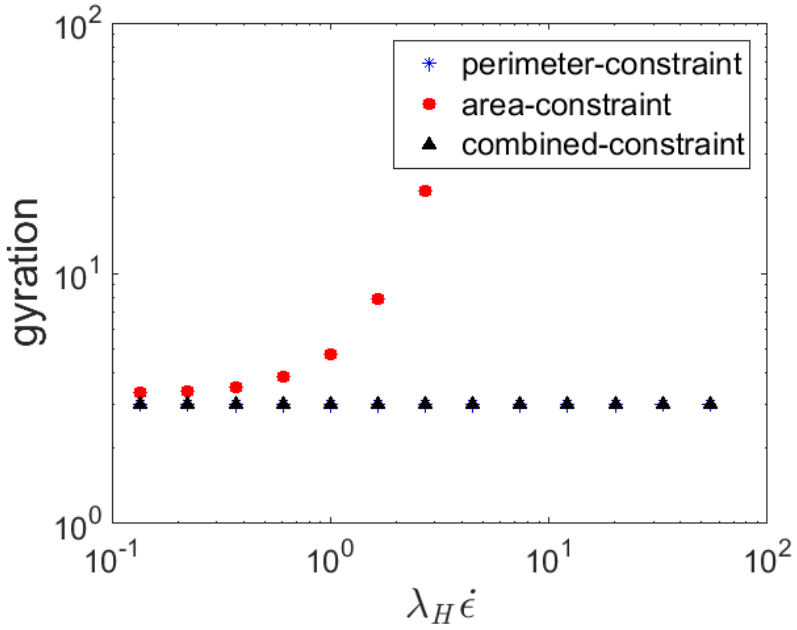


Figure 22. A comparison of gyration of the model as the elongation rate $\dot{\varepsilon}$ increases for the triangular model with constraints ($a = 100$).

Unlike the result of area only constrained model where the viscosity increases exponentially as the value of $\dot{\varepsilon}$ increases, the rate of increase in viscosity drops after certain point for combined-constraint model (Fig. 21). This is assumed because of decrease in degree of freedom in the constraint. The exponential growth in area-constraint model means the model can be stretched infinitely with Hookean spring which will not happen in reality. Therefore, the combined-constraint result gave an improvement to the model. Furthermore, when we do analysis of the possible shape of RBC, equilateral triangle was unlikely for two constraint model.

5.3 Bead-Spring Tetrahedron Model with Two Constraints

This microstructure model is tested in steady shear flow condition. Below is the result of viscosity and first normal stress difference for tetrahedron model. Three different simulation is compared in one plot; RBC model with area constraint only, with volume constraint only, and combination of two constraints (Fig. 23).

Each constraint by itself observed shear thinning effect, whereas, viscosity of the combined constraint does not show much dependence on the shear rate. The tetrahedron model with combined constraint does not have flexibility to change its shape compared with previous study with triangular model due to the decrease in the degree of freedom. Simiar result has been observed for area-constraint only of this model as the perimeter-constraint only of the triangular model in shear flow. It tends to fold with no volume.

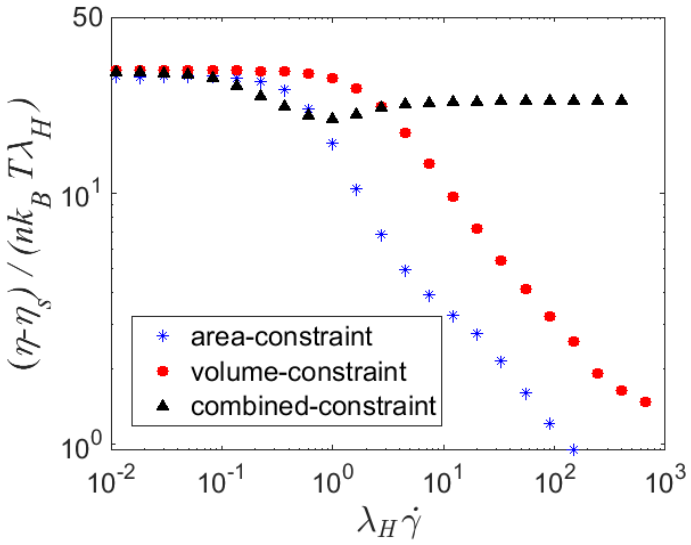


Figure 23. Comparison of viscosity plots of RBC tetrahedron model with constraint ($a' = 400$).

For first normal stress difference, all three simulation shows the dependence on the shear rate meaning that the constraint effect on the stress component xx and yy are not significant.

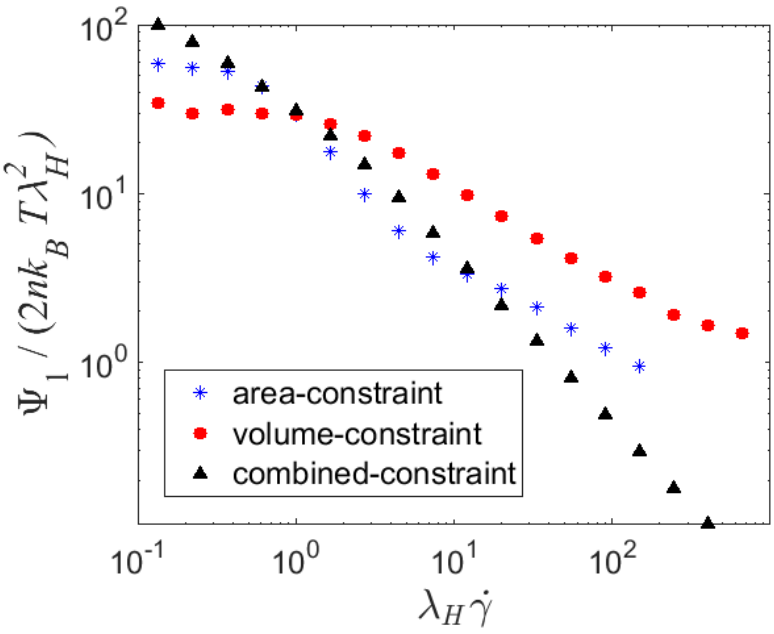


Figure 24. Comparison of first normal stress coefficient ψ_1 responses for the RBC tetrahedron model under constraint ($a' = 400$).

In general, we can verify that gyration is larger when large shear is applied to the system. Specifically, volume constraint deforms easier than the area constraint where the constraint is sum of area squared.

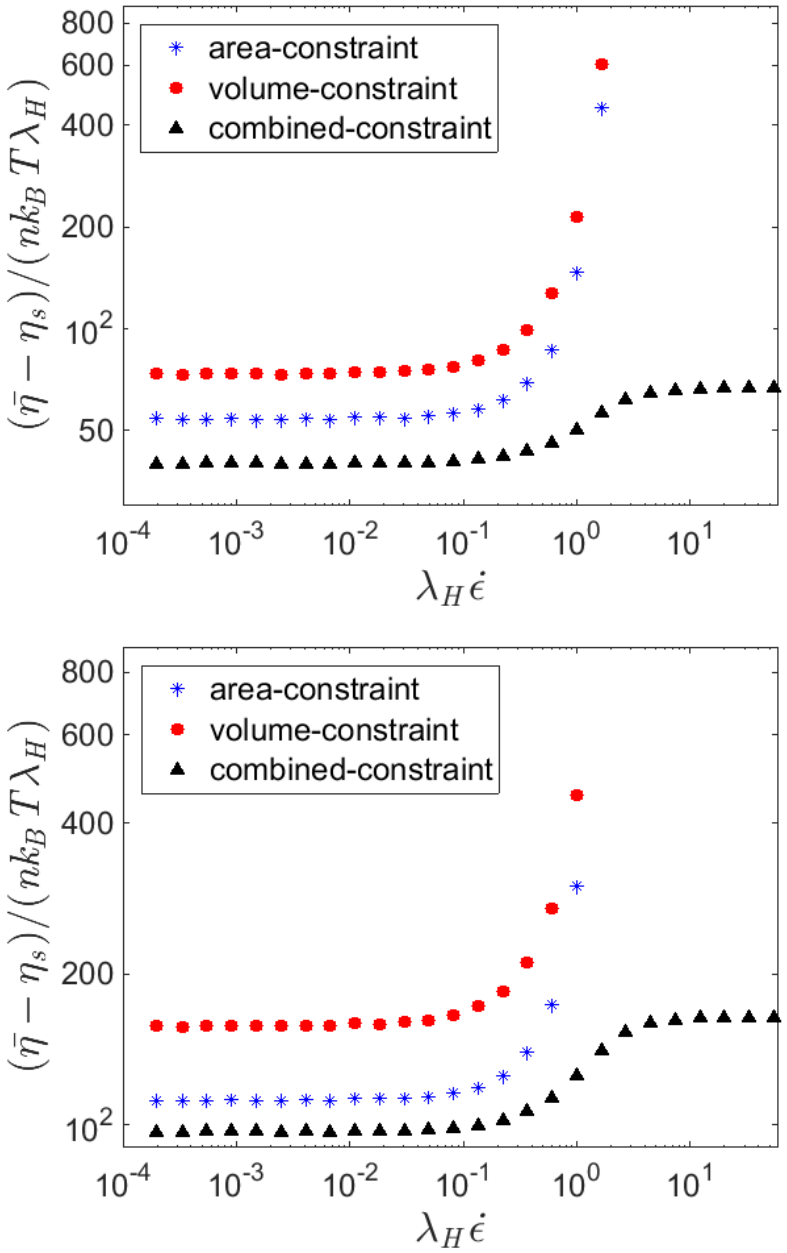


Figure 25. A comparison log-log plot of elongational viscosity $\bar{\eta}$ as a function of the elongation rate $\dot{\epsilon}$ for the tetrahedron model with constraints (a) $a' = 100$ and (b) $a' = 400$.

For single constraint, the elongational viscosity increases exponentially as the rate increases even with the volume only constraint due to the increase of degree of freedom (Fig. 25). When we see the gyration plot in Figure 26, it implies that the model stretches infinitely

which led to the result above for the elongational viscosity. In the combined constraint model, elongational rate reaches plateau state showing nearly constant value of gyration.

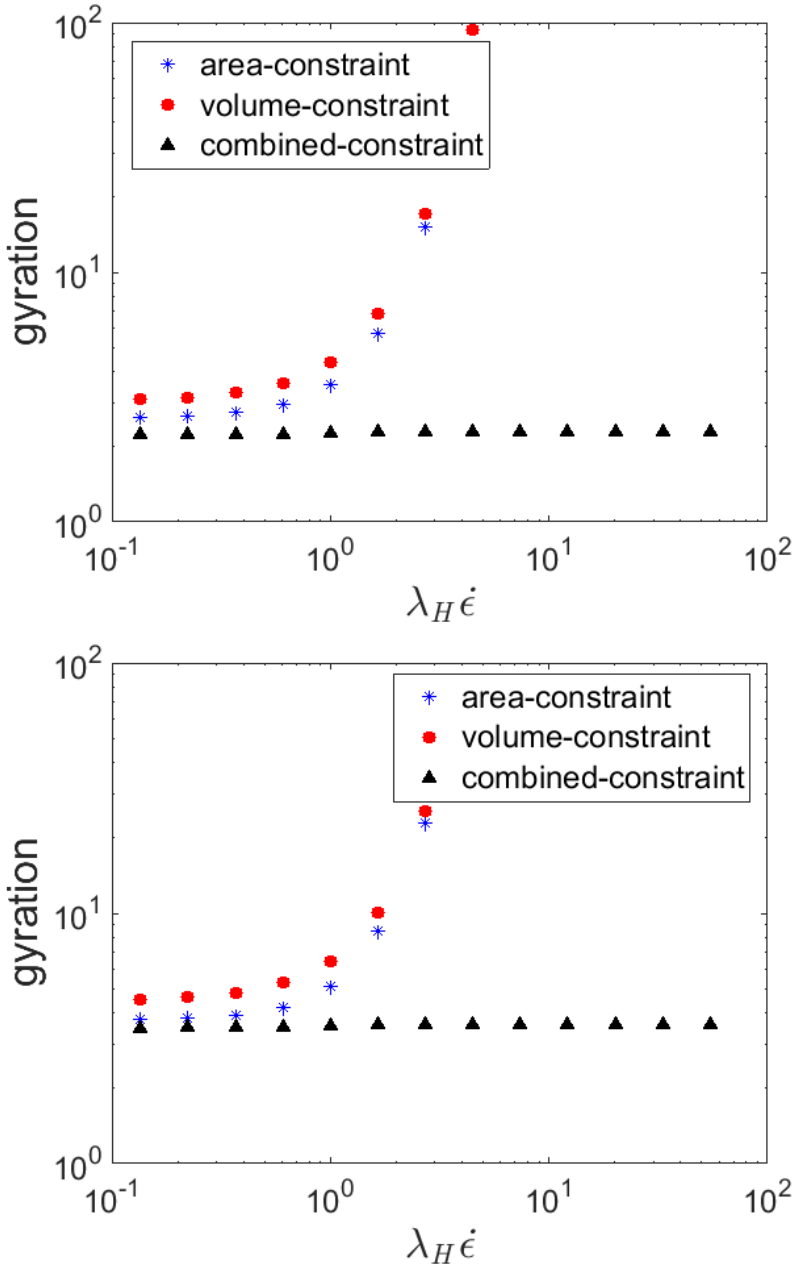


Figure 26. A comparison of gyration of the model as the elongation rate $\dot{\varepsilon}$ increases for the tetrahedron model with constraints (a) $a' = 100$ and (b) $a' = 400$.

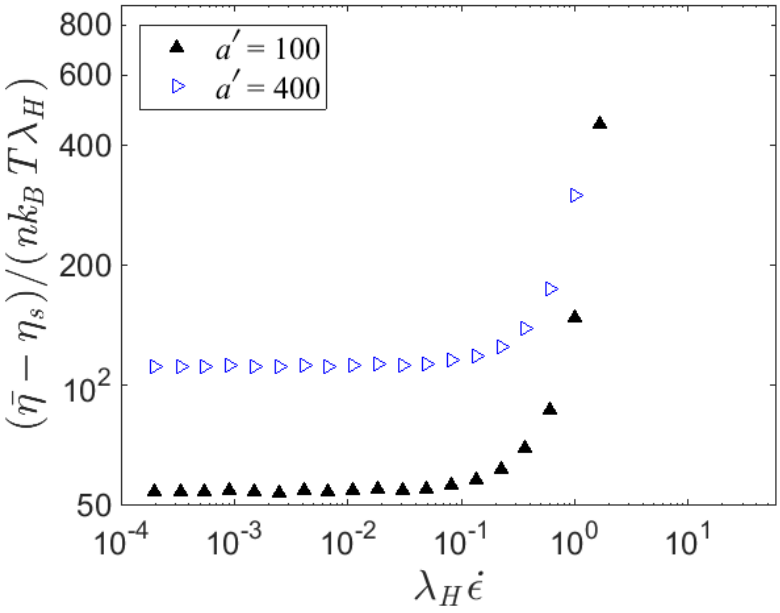


Figure 27. A comparison of elongational viscosity of single constraint tetrahedron model with different area sizes.

When we compare the result of area only constraint of the model with different area sizes, the zero rate viscosity of larger area size gives higher viscosity than the smaller area size.

5.4 Multi-Bead-Spring Model with One Constraint

The viscosity for Multi-Bead-Spring model with area constraint only is tested in steady shear flow condition. The tested sphere model contains 78 beads. There has been yet applied force to form biconcave shape of RBC. Result of viscosity started to show shear thinning effect after dimensionless shear rate of 1 where the decrease of the viscosity happens later than other models.

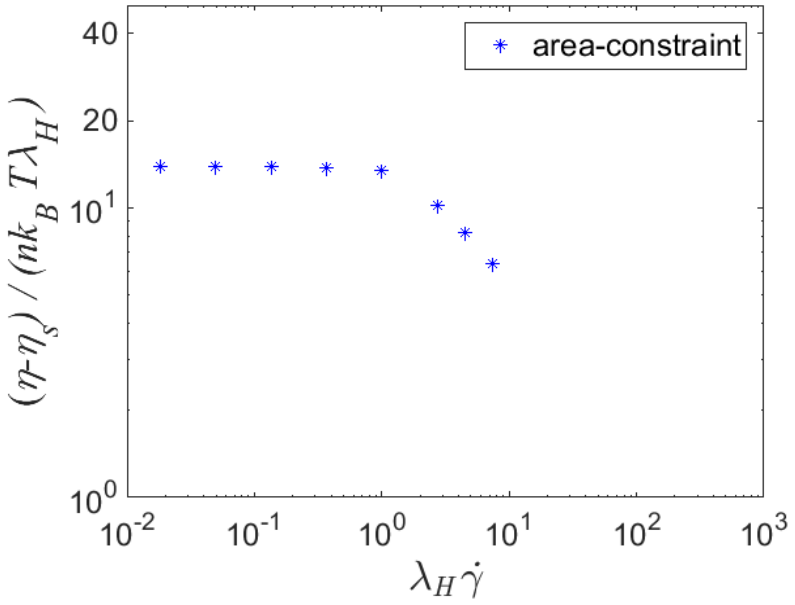


Figure 28. A viscosity plot of the Multi-Bead-Spring RBC model with respect to shear rate ($a'' = 100$).

Smaller value of Δt would be needed to retain the constraint to obtain the value in the high shear rate beyond dimensionless unit 10 in Figure 26. The model would retain its shape better with combined constraint.

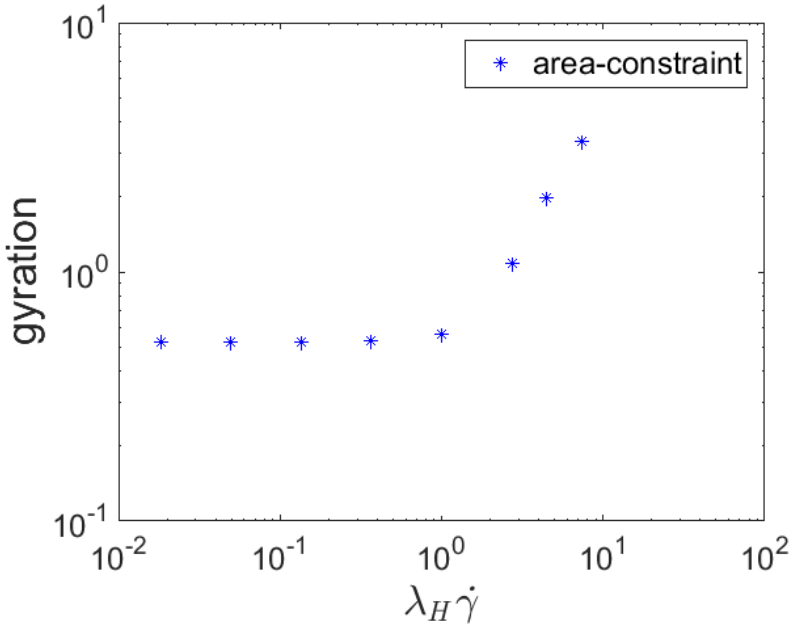


Figure 29. Gyration plot of the Multi-Bead-Spring RBC model with respect to shear rate ($a'' = 100$).

6. CONCLUSION AND DISCUSSION

The main developments here are the modeling microstructure of RBC and simulations of dilute solution of RBC model based on kinetic theory with constraint algorithm. Three different models we constructed and tested under shear and shear free flow. For the shear flow, stress growth upon inception of steady shear flow, steady shear flow, stress relaxation after a sudden shearing displacement, and steady shear flow in capillary is simulated. In the model developed part, a foundational framework is proposed.

In essence, the method outlined in this dissertation is an application of the study of polymeric fluid by Liu (1989) and Wiest (1987) to biorheological fluid. This procedure employed as a stress tensor calculator to study the motion of suspended RBC. Discrete models of the RBC are constructed using hydrodynamic resistant site and its connection to demonstrate its flow behavior. The constraints applied to the system allowed seeing the shearing dependence in the material function while maintain its overall size.

The findings of this study is that the Brownian Dynamic simulation results of the bead-spring ring with constraints show a powerful capability to model the rheological properties of red blood cell in steady flow and boundary under shear stress based on the shear thinning of viscosity, first normal tensor coefficient responses, and Fahraeus-Lindqvist effect. The rheological material properties for both single and combined constrained triangular models are presented and the results are qualitatively in agreement with experimental data (Copley, 1973; Dintenfass, 1974; Windberger, 2010) clearly showing shear-thinning effect of viscosity in shear flow. Adjusted parameters in this study are relaxation time, correlation between the size of the RBC and the slope.The model using both constraints has less degree of freedom giving more restriction to cell deformation; therefore, lead to less shear thinning of shear

properties than single constraint model where the constraint is only on area, perimeter, or volume. The short time for the stress relaxation after shear displacement explains the strong effect of the constraint forces on the RBC. We also observed Fahraeus-Lindqvist effect in the capillary flow with this study. The results involve the phenomenon of axial accumulation of red cells as well. In steady elongational flow, the rings remain in equilibrium at low elongation rates. At higher elongation rates, the ring is oriented along the axis of elongation and extended. The degree of extension and the viscosity increases with increasing elongation rate. It showed more realistic behavior in the elongational flow with combined-constraint model.

Overall, introducing another constraint to the model with only area constraint gave an improvement to the model. It was successful to see that the second constraint gave restriction to deformation while the model still has some degree of flexibility. For multi-bead model, degree of freedon would increase so that we are expected to see more shear thinning effect in the combined constraint model for shear flow. We therefore, conclude that using a method based on kinetic theory with holonomic constraints to simulate discrete bead-spring model of the RBC has shown the potential for future studies of blood of rheology.

The use of random variables in the simulation and a finite number of trajectories in the ensemble means that there is intrinsic statistical noise to the method that is used in this study (Doyle *et al*., 2005). The size of this error is proportional to the number of independent trajectories. There is a technique called variance reduction for the reduction of this error by reducing the proportionality factor rather than increasing the number of trajectories even more (Ö ttinger, 1996). Although the exact way of performing variance reduction depends on the system of study, we can investigate this to see if this can be done in our system. Moreover, the

simulation code can be further revised that the vector and tensor calculations are done in efficient way by using existing algebra packages, for example, since it is computationally intensive.

The further development of modeling RBC is extended to biconcave shape as shown in Fig. 1(a) with total surface area. The 3D mesh generator was beneficial to develop framework for N number of beads in the RBC structure to systematically find neighboring beads and obtain inter-particle forces. Moreover, the total surface area is easily calculated with the indices of each triangle from the generator.

There can be improvement to this research for future works. Some ideas are listed below. At this point, we need to first develop volume constraint to test combine constraint for biconcave RBC model. Setting up the constraint equation so that the constraint σ is differentiable by the bead position is inevitable to use the constraint method used in this study. We can consider using Gauss Divergence theorem using the curve of the surface. Rathod (2007) proposed a numerical integration algorithm of an arbitrary tetrahedron in three-dimensional space by summing four integrals of such arbitrary function over the unit triangle which can be tested as well. A good overview of methods for evaluating volume integrals can be found in Lee *et al*. (1982).

Second, we can investigate bending potential energy function to the springs connected to the bead in order to maintain a biconcave curve in the mid region of the RBC model. Representation of such is drawn below in Figure 30.

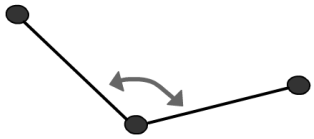


Figure 30. Schematic of a bending potential to the springs connected to the bead.

Similarly, the center of mass of RBC $\boldsymbol{r}_c(x_c, y_c, z_c)$ or the local center can be used to determine the direction of the local curvature to keep the biconcave shape. The local surface dotted with local center of mass will give the direction of the curve as shown in Figure 31.

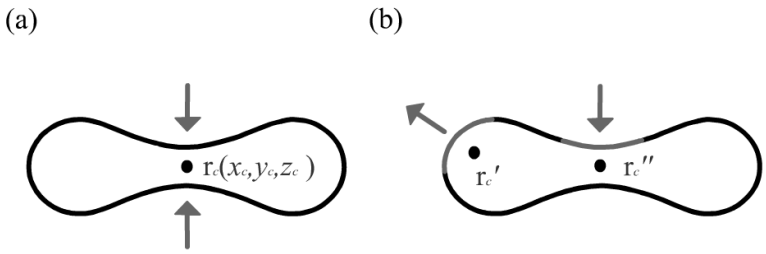


Figure 31. Schematic of using (a) center of mass of RBC $\boldsymbol{r_c(x_c, y_c, z_c)}$ (b) local center of mass to maintain the curve in the biconcave model.

As the model is build up, we expect the degree of freedom will increase so that the motion of RBC model will lead to result giving the shear-thinning effect that lines better with the experimental result. The use of the procedure can be extended to more complex situations and more realistic models to understand changes in flow behavior and rheological properties of RBC flow in small diameter vessels, and replacing the single vessel for more complex geometries (walls, constriction, bends, junction, networks) or combinations. In addition, this method makes feasible its use as a model capable of predicting the performance of RBC circulation over a wide range of physiological conditions of the RBC. Ultimately, our model

combined with aspects of other models may lead the best representation of RBC, and include interaction between blood cells.

CITED LITERATURE

Andersen, H.C.: Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations. *J. Comput. Phys*., 52(1): 24-34, 1983.

Atkinson, J., Goh, C.J., and Phan-Thien, N.: Bead-spring models for an adsorbed polymer molecule in a shear flow. *J. Chem. Phys.*, 80: 6305, 1984.

Bagchi, P.: Mesoscale Simulation of Blood Flow in Small Vessels. *Biophys. J*., 92: 1858–1877, 2007.

Biller, P. and Petruccione, F.: Rheological properties of polymer dumbbell models with configuration-dependent anisotropic friction. *J. Chem. Phys.*, 89: 2412, 1988.

Bird, R.B., Curtis, C.F., Armstrong, R.C., and Hassager, O.: *Dynamics of Polymeric Liquids*: *Volume 2 Kinetic Theory*. 2nd edition, New York, Wiley-Interscience, 1987.

Charm, S.E. and Kurland, G.S.: *Blood Flow and Microcirculation*. New York, John Wiley & Sons, 1974.

Ciccotti, G. and Ryckaert, J.P.: Molecular dynamics simulations of rigid molecules. *Comput. Phys. Rep.*, 4: 345-392, 1986.

Coates T.D.: Clinical Introduction to Sickle Cell Anemia and the Importance of Blood Rheology. *Biorheology*, 45: 85, 2008.

Copley, A.L.: On biorheology. *Biorheology*, 10(2): 87-105, 1973.

Cordoba, A., Indei, T. and Schieber J.D.: Elimination of Inertia from a Generalized Langevin Equation: Applications to Microbead Rheology Modeling and Data Analysis. *J. Rheol.*, 56: 185-212, 2012.

Davit, Y. and Peyla, P.: Intriguing Viscosity Effects in Confined Suspensions: A Numerical Study. *EPL*, 83: 64001, 2008.

Dintenfass, L.: Blood viscosity in healthy men, measured in rhombospheroid viscometer on EDTA blood. A comparison with the cone-in-cone viscometer data and those of Copley. *Biorheology*, 11: 397-403, 1974.

Dintenfass, L.: *Blood viscosity, Hyperviscosity and Hyperviscosaemia*. Boston, MTP, 1985.

Discher, D.E., Mohandas, N., and Evans, E.A.: Molecular maps of red cell deformation: Hidden elasticity and in situ connectivity. *Science*, 266: 1032, 1994.

Dotson, P.J.: Brownian dynamics simulation of macromolecules in steady shear flow. *J. Chem. Phys.*, 79: 5730, 1983.

Doyle, P.S and Underhill, P.T.: Brownian Dynamics Simulations of Polymers and Soft Matter. In: *Handbook of Materials Modeling*. Springer, 2005.

Doyle, P.S.: Brownian Dynamics Simulations of Polymers and Soft Matter. Retrieved from http://web.mit.edu/doylegroup/pubs/BD-Handbook-v5.pdf, 2014.

Evans, E. and Fung Y.: Improved measurements of the erythrocyte geometry. *Microvasc. Res.*, 4: 335-347, 1972.

Evans, E.A. and Hochmuth, R.M.: Membrane viscoelasticity. *Biophys. J.*, 16: 1-11, 1976.

Fahraeus, R. and Lindqvist, T.: The viscosity of the blood in narrow capillary tubes. *Am. J. Physiol.*, 96: 562-568, 1931.

Galdi, G.P., Rannacher, R., Robertson, M., and Turek, S.: *Hemodynamical Flows: Modeling, Analysis and Simulation*. Oberwolfach Seminars, vol 37, 2008.

Gidaspow, D. and Huang, J.: Kinetic Theory Based Model for Blood Flow and its Viscosity. *Ann. Biomed. Eng.*, 37(8): 1534-1545, 2009.

Gilmer, G. and Yip, S.: Basic Monte Carlo Models: Equilibrium and Kinetics. In: *Handbook of Materials Modeling*. Springer, 2005.

Goldsmith, H.L., Fed. Proc. 26: 1813-1820, 1967.

Haynes, R.H. and Burton, C.: Role of the non-Newtonian behavior of blood in hemodynamics. *J. Appl. Physiol*., 197: 943-950, 1959.

Healy, C. and Joly, M.: Rheological Behavior of Blood in Transient Flow. *Biorheology*, 12: 335-340, Figure 10, 1975.

Hosseini, S.M. and Feng, J.J.: A particle-based model for the transport of erythrocytes in capillaries. *Chem Eng. Sci.*, 64: 4488-4497, 2009.

Hosseini, S.M. and Feng, J.J.: A particle-based model for the transport of erythrocytes in capillaries. *Chem Eng. Sci.*, 64: 4488-4497, 2009.

Jorgensen, W. and Tirado-Rives, J.: Monte Carlo vs Molecular Dynamics for Conformational Sampling. *J. Phys. Chem.*, 100: 14508-14513, 1996.

Khodabandelou, T., Boisseau, M.R., and LeDevehat, C.: Blood Rheology as a Marker of Venous Hypertension in Patients with Venous Disease. *Clin. Hemorheol. Microcirc.*, 30: 307-312, 2004.

Kirkwood, J.G.: *Macromolecules*. New York, Gordon and Breach, 1967.

Kuchel, P.W., Fackerell, E.D.: Parametric-equation representation of biconcave erythrocytes. *Bull. Math. Biol.*, 61(2): 209-220, 1999.

Larkin, T.J., Kuchel, P.W.: Mathematical models of naturally "morphed" human erythrocytes: stomatocytes and echinocytes. *Bull. Math. Biol.*, 72(6):1323-1333, 2010.

Lee, Y.T. and Requicha, A.G.G.: Algorithms for computing the volume and other integral properties of solids I: known methods and open issues. *Commun. ACM*, 25: 635-641, 1982.

Leschke, M.: Rheological Factors in Coronary Heart Disease. *Deutsche Medizinishe Wochenschrift*, 133: S270-S273, 2008.

Liu, T.: Flexible polymer chain dynamics and rheological properties in steady flows. *J. Chem. Phys*., 90(10): 5826, 1989.

Liu, S-C., Derick, L., and Palek, J.: Visualization of the hexagonal lattice in the erythrocyte membrane skeleton. *J. CellBiol.*., 104: 527-536, 1987.

Liu, Y. and Liu, W.K.: Rheology of Red Blood Cell Aggregation by Computer Simulation. *J. Comput. Phys.*, 220: 139–154, 2006.

Lockhart, C.J., Hamilton, P.K., McVeigh, K.A., and McVeigh, G.E.: A Cardiologist View of Vascular Disease in Diabetes. *Diabetes Obes. Metab.*, 10: 279-292, 2008.

Lopez, R.H.: Blood Rheology using a Brownian Dynamics Simulation of Bead-Spring Rings with a Constant Area Constraint. PhD dissertation, University of Illinois at Chicago, Illinois, 2007.

Meller, J.: Molecular Dynamics. *Enc. Life Sci*., 1-8, 2001.

Merrill, E.W., Gilliland, E.R., Cokelet, G., Shin, H., Britten, A., and Wells, R.E.: Rheology of blood and flow in the microcirculation. *J. Appl. Physiol*., 18: 225, 1963.

Moyers-Gonzalez, M., Owens, R.G., and Fang, J.: A Non-Homogeneous Constitutive Model for Human Blood. Part 1. Model Derivation and Steady Flow. *J. Fluid Mech.*, 617: 327–354, 2008.

Novacek, G., Vogelsang, H., Genser, D., Moser, G., Gangl, A., Ehringer, H., and Koppensteiner, R.: Changes in Blood Rheology Caused by Crohn's Disease. *Euro. J. Gastroenterol. Hepatol.*, 8: 1089-1093, 2008.

Ö ttinger, H.C.: On the distribution of included angles for the Rouse chain in strong steady shear flow. *J. Chem. Phys.*, 84: 1850, 1986.

Ö ttinger, H.C.: Models with Constraints. In: *Stochastic Processes in Polymeric Fluids*. Springer, 1996.

Pan, T. and Wang, T.: Dynamical Simulation of Red Blood Cell Rheology in Microvessels. *Int. J. Numer. Anal. Model.*, 6(3): 455–473, 2009.

Persson, P.-O.: Mesh Generation for Implicit Geometries. PhD dissertation, Massachusetts Institute of Technology, 2005.

Peyla, P.: Rheology and Dynamics of a Deformable Object in a Microfluidic Configuration: A Numerical Study. *EPL*, 80: 34001, 2007.

Pozrikidis, C.: Axisymmetric motion of a file of red blood cells through capillaries. *Phys. Fluids*, 17: 031503, 2005.

Pozrikidis, C.: Numerical Simulation of the Flow-Induced Deformation of Red Blood Cells. *Ann. Biomed. Eng.*, 31: 1194, 2003.

Pries, A.R., Neuhaus D., Gaehtgens P.: Blood viscosity in tube flow: dependence on diameter and hematocrit. *Am. J. Physiol. Heart Circ. Physiol.*, 263(6): H1770-H1778, 1992.

Pries, A.R. and Secomb, T.W.: Rheology of the Microcirculation. *Clin. Hemorheol. Microcirc.*, 29:143–148, 2003.

Puig-De-Morales-Marinkovic, M., Turner, K.T., Butler, J.P., Fredberg, J.J., and Suresh, S.: Viscoelasticity of the Human Red Blood Cell. *Am. J. Physiol. Cell Physiol.*, 293:597–605, 2007.

Qin, X.: Report on Continued Simulation of three Bead-Spring Rings with a Constant Area Constraint. University of Illinois at Chicago, Illinois, 2008.

Rathod, H.T., Nagaraja, K.V., Venkatesudu, B.: Numerical integration of some functions over an arbitrary linear tetrahedra in Euclidean three-dimensional space. *Appl. Math. Comput.*, 191:397-409, 2007.

Risken, H.: *The Fokker-Planck Equation: Methods of Solution and Applications*. 2nd edition, Berlin, Springer, 1989.

Ryckaert, J.P., Ciccotti, G., and Berendsen, H.J.C.: Numerical integration of the Cartesian equations of motion of a system with constaints: molecular dynamics of n-alkanes. *J. Comput. Phys*., 23:327-341, 1977.

Saab, H.H. and Dotson, P.J.: Nonequilibrium statistics of flexible macromolecules in dilute solutions. II. Macromolecular extension and comparison with nonequilibrium Brownian dynamics. *J. Chem. Phys.*, 86: 3039, 1987.

Secomb, T.W.: *Modeling and Simulation of Capsules and Biological Cells*. edited by Pozrikidis, C., Boca Raton, Chapman & Hall/CRC, 2003.

Secomb, T.W.: Red Blood Cell Mechanics and Capillary Blood Rheology. *Cell Biophysics* 18: 231, 1991.

Schieber, J.D. and Ö ttinger, H.C.: The effects of bead inertia on the Rouse model. *J. Chem.Phys.*, 89 (11): 6972-6981, 1988.

Sung, L.A. and Vera, C.: Protofilament and Hexagon: A Three-Dimensional Mechanical Model for the Junctional Complex in the Erythrocyte Membrane Skeleton. *Ann. of Biomed. Eng.*, 31: 1314-1326, 2003.

Snyder, G.K. and Sears, R.D.: Red Blood Cell Size and the Fahraeus–Lindqvist Effect. *Can. J. Zool.*, 84: 419–424, 2006.

Thurston, G.B., Henderson, N.M. and Jeng, M.: Effects of erythrocytapheresis transfusion on the viscoelasticity of sickle cell blood. *Clin. Hemorheol. Microcirc.*, 30: 61-75, 2004.

Tsubota, K.: Particle method for computer simulation of red blood cell motion in blood flow. *Comput. Meth. Prog. Biomed*., 83(2): 139-146, 2006.

Tsukada, K.: Direct measurement of erythrocyte deformability in diabetes mellitus with a transparent microchannel capillary model and high-speed video camera system. *Microvasc. Res*., 61:231-239, 2001.

Valencia, A., Morales, H., Rivera, R., Bravo, E., and Galvez, M.: Blood Flow Dynamics In Patient-Specific Cerebral Aneurysm Models: The Relationship Between Wall Shear Stress And Aneurysm Area Index. *Med. Eng. & Phys*., 30: 329-340, 2008.

Wedgewood, Lewis E.: An objective rotation tensor applied to non-Newtonian fluid mechanics. *Rheologica Acta*, 38: 91-99, 1999.

Wiest, J.M., Burdette, S.R., Liu, T.W., and Bird, R.B.: Effect of ring closure on rheological behavior. *J. Non-Newton. Fluid Mech.*, 24: 279-295, 1987.

Windberger, U.: Whole blood viscosity, plasma viscosity and erythrocyte aggregation in nine mammalian species: reference values and comparison of data. *Exp. Physiol.*, 88.3: 431–440, 2010.

Wolstenholme, G.E.W., Knight, J., and Goldsmith, H.L.: *Circulatory and respiratory mass transport: a Ciba Foundation symposium.* London, Churchill: Ciba Foundation, 1969.

Zhou, R. and Chang, H-C.: Capillary Penetration Failure of Blood Suspensions. *J. Colloid Interface Sci.*, 287: 647–656, 2005.

APPENDICES

Appendix A
Distribution Function and Definition of its Average

The distribution function in position-velocity space can be expressed as configuration-space distribution function and velocity-space distribution function.

$$F(\boldsymbol{r}_\nu, \dot{\boldsymbol{r}}_\nu, t) = \Psi(\boldsymbol{r}_\nu, t)\Xi(\dot{\boldsymbol{r}}_\nu, \boldsymbol{r}_\nu, t) \quad \text{(A-1)}$$

To indicate that the distribution of configurations is independent of the location of the particle in space, we can factor configuration-space distribution as following.

$$\Psi(\boldsymbol{r}_\nu, t) = n\psi(\boldsymbol{Q}, t) \quad \text{(A-2)}$$

Per unit volume, there are $\boldsymbol{n}$ polymer molecules. The distribution function $\boldsymbol{\psi}$ satisfies the normalization condition of

$$\int \psi(\boldsymbol{Q}, t) d\boldsymbol{Q} = 1 \quad \text{(A-3)}$$

When we assume the distribution to be Maxwellian, the velocity-space distribution function is

$$\Xi_{\text{eq}}(\dot{\boldsymbol{r}}_\nu) = \frac{\exp\left\{-\sum_1^\nu \left[\frac{1}{2}m(\dot{\boldsymbol{r}}_i - \boldsymbol{v})^2\right]/k_B T\right\}}{\int_{-\infty}^{+\infty} \dots \int_{-\infty}^{+\infty} \exp\left\{-\sum_1^\nu \left[\frac{1}{2}m(\dot{\boldsymbol{r}}_i - \boldsymbol{v})^2\right]/k_B T\right\} d\dot{\boldsymbol{r}}_1 \dots d\dot{\boldsymbol{r}}_\nu} \quad \text{(A-4)}$$

where is the mass-average velocity of the solution and satisfies the normalization condition of

$$\int \dots \int \Xi d\dot{\boldsymbol{r}}_1 \dots d\dot{\boldsymbol{r}}_\nu = 1 \quad \text{(A-5)}$$

The assumption made here is that velocity distribution in a flow system is the same as that in a solution at equilibrium. This is only used in the Brownian motion term in the equation and in the bead-momentum flux contribution to the stress tensor (Bird *et al.* 1987).

Average values

A velocity-space average of a time-independent function $B(\boldsymbol{r}_\nu, \dot{\boldsymbol{r}}_\nu)$ is

$$[\![B]\!] = \int \dots \int B\Xi d\dot{\boldsymbol{r}}_1 \dots d\dot{\boldsymbol{r}}_\nu \quad \text{(A-6)}$$

where the velocity of the bead is $\dot{\boldsymbol{r}}_\nu = d\boldsymbol{r}_\nu/dt$ with respect to the same origin. The phase-space average in a function of time $\boldsymbol{t}$ is then

$$\langle B\rangle = \frac{1}{nV}\int \ldots \int [\![B]\!]\Psi d\boldsymbol{r}_1 \ldots d\boldsymbol{r}_\nu \tag{A-7}$$

If B depends only on $\boldsymbol{Q}$, then

$$\langle B\rangle = \int B\psi(\boldsymbol{Q},t)d\boldsymbol{Q} \tag{A-8}$$

The rest of the components have been proven to be zero for incompressible fluids.

Appendix B
Gradient of Constraints

The definition of constraints and its derivatives are derived below for each model of microstructure. The notation used in this dissertaion is that of Bird *et al.* (1987).

For Triangle Model,

Area constraint is

$$\sigma_1(\{\boldsymbol{r}\}) = \frac{1}{4}(|\boldsymbol{Q}_1 \times \boldsymbol{Q}_2| \cdot |\boldsymbol{Q}_1 \times \boldsymbol{Q}_2|) - a^2 = 0 \qquad \text{(B-1)}$$

Sum of length square constraint is

$$\sigma_2(\{\boldsymbol{r}\}) = (\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_1 + \boldsymbol{Q}_2 \cdot \boldsymbol{Q}_2 + \boldsymbol{Q}_3 \cdot \boldsymbol{Q}_3) - l^2 = 0 \qquad \text{(B-2)}$$

where the connector vector is defined as

$$\boldsymbol{Q}_1 = \boldsymbol{r}_2 - \boldsymbol{r}_1, \qquad \boldsymbol{Q}_2 = \boldsymbol{r}_3 - \boldsymbol{r}_1, \qquad \boldsymbol{Q}_3 = \boldsymbol{r}_3 - \boldsymbol{r}_2 \qquad \text{(B-3)}$$

For simplicity, we let $\boldsymbol{w} = \boldsymbol{Q}_1 \times \boldsymbol{Q}_2$, then the gradients of the constraints are

$$\nabla\sigma_1 = \frac{1}{4}\nabla(\boldsymbol{w} \cdot \boldsymbol{w}) \ = \frac{1}{2}[\nabla\boldsymbol{w} \cdot \boldsymbol{w}] \qquad \text{(B-4)}$$

$$\nabla\sigma_2 = 2[\nabla\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_1 + \nabla\boldsymbol{Q}_2 \cdot \boldsymbol{Q}_2 + \nabla\boldsymbol{Q}_3 \cdot \boldsymbol{Q}_3] \qquad \text{(B-5)}$$

using the identity for vector calculation.

The term $\nabla\boldsymbol{w}$ in (B-4) is

$$\nabla\boldsymbol{w}_\nu = \{\nabla\boldsymbol{Q}_1 \times \boldsymbol{Q}_2 - \nabla\boldsymbol{Q}_2 \times \boldsymbol{Q}_1\}, \qquad \nu = 1, 2, 3 \qquad \text{(B-6)}$$

Solving this for each bead gives

$$\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{r}_1} = -\underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_2 + \underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_1 \qquad \text{(B-6a)}$$

$$\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{r}_2} = \underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_2 \qquad \text{(B-6b)}$$

$$\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{r}_3} = -\underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_1 \qquad \text{(B-6c)}$$

The term $\nabla \boldsymbol{Q}$ in (B-5) gives

$$\frac{\partial \boldsymbol{Q}_1}{\partial \boldsymbol{r}_1} = -\underline{\underline{\boldsymbol{\delta}}}, \qquad \frac{\partial \boldsymbol{Q}_2}{\partial \boldsymbol{r}_1} = -\underline{\underline{\boldsymbol{\delta}}}, \qquad \frac{\partial \boldsymbol{Q}_3}{\partial \boldsymbol{r}_1} = 0, \tag{B-7}$$

$$\frac{\partial \boldsymbol{Q}_1}{\partial \boldsymbol{r}_2} = \underline{\underline{\boldsymbol{\delta}}}, \qquad \frac{\partial \boldsymbol{Q}_2}{\partial \boldsymbol{r}_2} = 0, \qquad \frac{\partial \boldsymbol{Q}_3}{\partial \boldsymbol{r}_2} = -\underline{\underline{\boldsymbol{\delta}}},$$

$$\frac{\partial \boldsymbol{Q}_1}{\partial \boldsymbol{r}_3} = 0\ , \qquad \frac{\partial \boldsymbol{Q}_2}{\partial \boldsymbol{r}_3} = \underline{\underline{\boldsymbol{\delta}}}, \qquad \frac{\partial \boldsymbol{Q}_3}{\partial \boldsymbol{r}_3} = \underline{\underline{\boldsymbol{\delta}}}$$

By substituting all (B-6a) through (B-7) into (B-4) and (B-5) gives the gradient of constraints with respect to position of the beads.

$$\frac{\partial \sigma_1}{\partial \boldsymbol{r}_1} = \frac{1}{2}\left(\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{r}_1} \cdot \boldsymbol{w}\right) = \frac{1}{2}\left(-\underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_2 + \underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_1\right) \cdot (\boldsymbol{Q}_1 \times \boldsymbol{Q}_2) \tag{B-8a}$$

$$= \frac{1}{2}[\boldsymbol{Q}_1(\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2 - \boldsymbol{Q}_2 \cdot \boldsymbol{Q}_2) + \boldsymbol{Q}_2(\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2 - \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_1)]$$

$$\frac{\partial \sigma_1}{\partial \boldsymbol{r}_2} = \frac{1}{2}\left(\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{r}_2} \cdot \boldsymbol{w}\right) = \frac{1}{2}\left(\underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_2\right) \cdot (\boldsymbol{Q}_1 \times \boldsymbol{Q}_2) \tag{B-8b}$$

$$= \frac{1}{2}[\boldsymbol{Q}_1(\boldsymbol{Q}_2 \cdot \boldsymbol{Q}_2) - \boldsymbol{Q}_2(\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2)]$$

$$\frac{\partial \sigma_1}{\partial \boldsymbol{r}_3} = \frac{1}{2}\left(\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{r}_3} \cdot \boldsymbol{w}\right) = \frac{1}{2}\left(-\underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_1\right) \cdot (\boldsymbol{Q}_1 \times \boldsymbol{Q}_2) \tag{B-8c}$$

$$= \frac{1}{2}[-\boldsymbol{Q}_1(\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2) + \boldsymbol{Q}_2(\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_1)]$$

$$\frac{\partial \sigma_2}{\partial \boldsymbol{r}_1} = -2\left(\underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_1 + \underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_2\right) = -2[\boldsymbol{Q}_1 + \boldsymbol{Q}_2] \tag{B-9a}$$

$$\frac{\partial \sigma_2}{\partial \boldsymbol{r}_2} = 2\left(\underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_1 - \underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_3\right) = 2[\boldsymbol{Q}_1 - \boldsymbol{Q}_3] \tag{B-9b}$$

$$\frac{\partial \sigma_2}{\partial \boldsymbol{r}_3} = 2\left(\underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_2 + \underline{\underline{\boldsymbol{\delta}}} \cdot \boldsymbol{Q}_3\right) = 2[\boldsymbol{Q}_2 + \boldsymbol{Q}_3] \tag{B-9c}$$

These gradients in (B-8a) through (B-8c) and (B-9a) through (B-9c) are used in the constraint subroutine in the computer simulations.

For Tetrahedron Model,

Area constraint is defined as

$$\sigma_1(\{\boldsymbol{r}\}) = \frac{1}{4}(|\boldsymbol{Q}_1 \times \boldsymbol{Q}_2|^2 + |\boldsymbol{Q}_2 \times \boldsymbol{Q}_3|^2 + |\boldsymbol{Q}_3 \times \boldsymbol{Q}_1|^2 + |\boldsymbol{Q}_4 \times \boldsymbol{Q}_5|^2) - a' = 0 \tag{B-10}$$

Volume constraint is

$$\sigma_2(\{\boldsymbol{r}\}) = \frac{1}{6}(\boldsymbol{Q}_3 \cdot [\boldsymbol{Q}_1 \times \boldsymbol{Q}_2]) - b = 0 \tag{B-11}$$

where the connector vectors are redefined for tetrahedron model.

$$\boldsymbol{Q}_1 = \boldsymbol{r}_2 - \boldsymbol{r}_1, \quad \boldsymbol{Q}_2 = \boldsymbol{r}_3 - \boldsymbol{r}_1, \quad \boldsymbol{Q}_3 = \boldsymbol{r}_4 - \boldsymbol{r}_1, \tag{B-12}$$
$$\boldsymbol{Q}_4 = \boldsymbol{r}_3 - \boldsymbol{r}_2, \quad \boldsymbol{Q}_5 = \boldsymbol{r}_4 - \boldsymbol{r}_2, \quad \boldsymbol{Q}_6 = \boldsymbol{r}_4 - \boldsymbol{r}_3$$

For simplicity, we will introduce vector $\boldsymbol{w}$ for intermediate step.

$$\boldsymbol{w}_1 = \boldsymbol{Q}_1 \times \boldsymbol{Q}_2, \quad \boldsymbol{w}_2 = \boldsymbol{Q}_2 \times \boldsymbol{Q}_3, \quad \boldsymbol{w}_3 = \boldsymbol{Q}_3 \times \boldsymbol{Q}_1, \quad \boldsymbol{w}_4 = \boldsymbol{Q}_4 \times \boldsymbol{Q}_5 \tag{B-13}$$

The gradients can be expressed in terms of $\boldsymbol{w}$.

$$\nabla\sigma_1 = \frac{1}{4}[\nabla(\boldsymbol{w}_1 \cdot \boldsymbol{w}_1) + \nabla(\boldsymbol{w}_2 \cdot \boldsymbol{w}_2) + \nabla(\boldsymbol{w}_3 \cdot \boldsymbol{w}_3) + \nabla(\boldsymbol{w}_4 \cdot \boldsymbol{w}_4)]$$
$$= \frac{1}{2}[\nabla\boldsymbol{w}_1 \cdot \boldsymbol{w}_1 + \nabla\boldsymbol{w}_2 \cdot \boldsymbol{w}_2 + \nabla\boldsymbol{w}_3 \cdot \boldsymbol{w}_3 + \nabla\boldsymbol{w}_4 \cdot \boldsymbol{w}_4] \tag{B-14}$$

$$\nabla\sigma_2 = \frac{1}{6}\nabla[\boldsymbol{Q}_3 \cdot \boldsymbol{w}_1] = \frac{1}{6}[\nabla\boldsymbol{Q}_3 \cdot \boldsymbol{w}_1 + \nabla\boldsymbol{w}_1 \cdot \boldsymbol{Q}_3] \tag{B-15}$$

The $\nabla\boldsymbol{w}$ terms are

$$\nabla\boldsymbol{w}_{1,\nu} = \{\nabla\boldsymbol{Q}_1 \times \boldsymbol{Q}_2 - \nabla\boldsymbol{Q}_2 \times \boldsymbol{Q}_1\} \qquad \nu = 1,2,3,4 \tag{B-16a}$$

$$\nabla\boldsymbol{w}_{2,\nu} = \{\nabla\boldsymbol{Q}_2 \times \boldsymbol{Q}_3 - \nabla\boldsymbol{Q}_3 \times \boldsymbol{Q}_2\} \tag{B-16b}$$

$$\nabla\boldsymbol{w}_{3,\nu} = \{\nabla\boldsymbol{Q}_3 \times \boldsymbol{Q}_1 - \nabla\boldsymbol{Q}_1 \times \boldsymbol{Q}_3\} \tag{B-16c}$$

$$\nabla\boldsymbol{w}_{4,\nu} = \{\nabla\boldsymbol{Q}_4 \times \boldsymbol{Q}_5 + \nabla\boldsymbol{Q}_5 \times \boldsymbol{Q}_4\} \tag{B-16d}$$

Solving for each bead, we get

$$\frac{\partial \boldsymbol{w_1}}{\partial \boldsymbol{r_1}} = -\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_2 + \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_1, \qquad \frac{\partial \boldsymbol{w_2}}{\partial \boldsymbol{r_1}} = -\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_3 + \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_2, \tag{B-16e}$$

$$\frac{\partial \boldsymbol{w_3}}{\partial \boldsymbol{r_1}} = -\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_1 + \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_3, \qquad \frac{\partial \boldsymbol{w_4}}{\partial \boldsymbol{r_1}} = 0$$

$$\frac{\partial \boldsymbol{w_1}}{\partial \boldsymbol{r_2}} = \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_2\,, \qquad \frac{\partial \boldsymbol{w_2}}{\partial \boldsymbol{r_2}} = 0, \tag{B-16f}$$

$$\frac{\partial \boldsymbol{w_3}}{\partial \boldsymbol{r_2}} = -\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_3, \qquad \frac{\partial \boldsymbol{w_4}}{\partial \boldsymbol{r_2}} = -\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_5 + \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_4$$

$$\frac{\partial \boldsymbol{w_1}}{\partial \boldsymbol{r_3}} = -\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_1, \qquad \frac{\partial \boldsymbol{w_2}}{\partial \boldsymbol{r_3}} = \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_3\,, \tag{B-16g}$$

$$\frac{\partial \boldsymbol{w_3}}{\partial \boldsymbol{r_3}} = 0\,, \qquad \frac{\partial \boldsymbol{w_4}}{\partial \boldsymbol{r_3}} = \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_5$$

$$\frac{\partial \boldsymbol{w_1}}{\partial \boldsymbol{r_4}} = 0\,, \qquad \frac{\partial \boldsymbol{w_2}}{\partial \boldsymbol{r_4}} = -\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_2, \tag{B-16h}$$

$$\frac{\partial \boldsymbol{w_3}}{\partial \boldsymbol{r_4}} = \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_1, \qquad \frac{\partial \boldsymbol{w_4}}{\partial \boldsymbol{r_4}} = -\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_4$$

Similarly, calculating (B-14) and (B-15) gives

$$\frac{\partial \sigma_1}{\partial \boldsymbol{r_1}} = \frac{1}{2}\left[\nabla \boldsymbol{w_{1,1}} \cdot \boldsymbol{w_1} + \nabla \boldsymbol{w_{2,1}} \cdot \boldsymbol{w_2} + \nabla \boldsymbol{w_{3,1}} \cdot \boldsymbol{w_3} + \nabla \boldsymbol{w_{4,1}} \cdot \boldsymbol{w_4}\right] \tag{B-16}$$

$$= \frac{1}{2}\begin{bmatrix}\left(-\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_2 + \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_1\right) \cdot (\boldsymbol{Q}_1 \times \boldsymbol{Q}_2) + \left(-\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_3 + \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_2\right) \cdot (\boldsymbol{Q}_2 \times \boldsymbol{Q}_3) \\ + \left(-\underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_1 + \underline{\underline{\boldsymbol{\delta}}} \times \boldsymbol{Q}_3\right) \cdot (\boldsymbol{Q}_3 \times \boldsymbol{Q}_1) + 0 \cdot (\boldsymbol{Q}_4 \times \boldsymbol{Q}_5)\end{bmatrix}$$

$$= \frac{1}{2}\left[\boldsymbol{Q}_1(A_{1,1}) + \boldsymbol{Q}_2(A_{1,2}) + \boldsymbol{Q}_3(A_{1,3})\right]$$

Therefore, in general

$$\frac{\partial \sigma_1}{\partial \boldsymbol{r_\nu}} = \frac{1}{2}\left[\boldsymbol{Q}_1(A_{\nu,1}) + \boldsymbol{Q}_2(A_{\nu,2}) + \boldsymbol{Q}_3(A_{\nu,3}) + \boldsymbol{Q}_4(A_{\nu,4}) + \boldsymbol{Q}_5(A_{\nu,5})\right] \tag{B-17}$$

where $m = 1, 2, 3, 4, 5$ and the scalar components $A_{\nu,m}$ of vector $\nabla\sigma_1$ are grouped for simplicity.

$$A_{1,1} = -\boldsymbol{Q}_2 \cdot \boldsymbol{Q}_2 + \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2 + \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_3 - \boldsymbol{Q}_3 \cdot \boldsymbol{Q}_3 \tag{B-18a}$$
$$A_{1,2} = \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2 - \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_1 - \boldsymbol{Q}_3 \cdot \boldsymbol{Q}_3 + \boldsymbol{Q}_2 \cdot \boldsymbol{Q}_3$$
$$A_{1,3} = \boldsymbol{Q}_2 \cdot \boldsymbol{Q}_3 - \boldsymbol{Q}_2 \cdot \boldsymbol{Q}_2 - \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_1 + \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_3$$

$$A_{1,4} = 0$$
$$A_{1,5} = 0$$

$$A_{2,1} = \boldsymbol{Q}_2 \cdot \boldsymbol{Q}_2 + \boldsymbol{Q}_3 \cdot \boldsymbol{Q}_3 \quad \text{(B-18b)}$$
$$A_{2,2} = -\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2$$
$$A_{2,3} = -\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_3$$
$$A_{2,4} = -\boldsymbol{Q}_5 \cdot \boldsymbol{Q}_5 + \boldsymbol{Q}_4 \cdot \boldsymbol{Q}_5$$
$$A_{2,5} = \boldsymbol{Q}_4 \cdot \boldsymbol{Q}_5 - \boldsymbol{Q}_4 \cdot \boldsymbol{Q}_4$$

$$A_{3,1} = -\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2 \quad \text{(B-18c)}$$
$$A_{3,2} = \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_1 + \boldsymbol{Q}_3 \cdot \boldsymbol{Q}_3$$
$$A_{3,3} = -\boldsymbol{Q}_2 \cdot \boldsymbol{Q}_3$$
$$A_{3,4} = \boldsymbol{Q}_5 \cdot \boldsymbol{Q}_5$$
$$A_{3,5} = -\boldsymbol{Q}_4 \cdot \boldsymbol{Q}_5$$

$$A_{4,1} = -\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_3 \quad \text{(B-18d)}$$
$$A_{4,2} = -\boldsymbol{Q}_2 \cdot \boldsymbol{Q}_3$$
$$A_{4,3} = \boldsymbol{Q}_2 \cdot \boldsymbol{Q}_2 + \boldsymbol{Q}_1 \cdot \boldsymbol{Q}_1$$
$$A_{4,4} = -\boldsymbol{Q}_4 \cdot \boldsymbol{Q}_5$$
$$A_{4,5} = \boldsymbol{Q}_4 \cdot \boldsymbol{Q}_4$$

Therefore,

$$\nabla\sigma_2 = \frac{1}{6}[\nabla\boldsymbol{Q}_1(\boldsymbol{Q}_2 \cdot \boldsymbol{Q}_3) + \nabla\boldsymbol{Q}_2(\boldsymbol{Q}_3 \cdot \boldsymbol{Q}_1) + \nabla\boldsymbol{Q}_3(\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2)] \quad \text{(B-19)}$$

$$\frac{\partial\sigma_2}{\partial\boldsymbol{r}_1} = -\frac{1}{6}[(\boldsymbol{Q}_1 \times \boldsymbol{Q}_2) + (\boldsymbol{Q}_2 \times \boldsymbol{Q}_3) + (\boldsymbol{Q}_3 \times \boldsymbol{Q}_1)] \quad \text{(B-19a)}$$

$$\frac{\partial\sigma_2}{\partial\boldsymbol{r}_2} = \frac{1}{6}(\boldsymbol{Q}_2 \times \boldsymbol{Q}_3) \quad \text{(B-19b)}$$

$$\frac{\partial\sigma_2}{\partial\boldsymbol{r}_3} = -\frac{1}{6}(\boldsymbol{Q}_1 \times \boldsymbol{Q}_3) = \frac{1}{6}(\boldsymbol{Q}_3 \times \boldsymbol{Q}_1) \quad \text{(B-19c)}$$

$$\frac{\partial\sigma_2}{\partial\boldsymbol{r}_4} = \frac{1}{6}(\boldsymbol{Q}_1 \times \boldsymbol{Q}_2) \quad \text{(B-19d)}$$

For each constraint, it satisfies $\sum_{\nu}^{N} \frac{\partial\sigma}{\partial\boldsymbol{r}_\nu} = 0$.

For Multi-Bead Model,

We can generalize the form for N particles using the information of indices of each triangle and the position of all beads stored in workspaces from Appendix F. Then the area constraint is

$$\sigma_1(\{\boldsymbol{r}\}) = \frac{1}{4}\sum_m^{n\_tri}\left(\left|\boldsymbol{Q}_{\mathrm{m},1} \times \boldsymbol{Q}_{\mathrm{m},2}\right|^2\right) - a'' = 0 \tag{B-20}$$

where connector vectors are defined as following for each triangle using Table IV in Section 3.3.1. n_tri is the total number of triangles.

$$\boldsymbol{Q}_{m,1} = \boldsymbol{r}_{\mathrm{B}} - \boldsymbol{r}_A, \quad \boldsymbol{Q}_{m,2} = \boldsymbol{r}_{\mathrm{C}} - \boldsymbol{r}_A \tag{B-21}$$

For simplicity, we will introduce vector $\boldsymbol{w}$ for intermediate step.

$$\boldsymbol{w}_m = \boldsymbol{Q}_{\mathrm{m},1} \times \boldsymbol{Q}_{\mathrm{m},2}$$

Then, the gradients of the constraints are

$$\begin{aligned}\nabla\sigma_{1,\nu} &= \frac{1}{4}\sum_m^{n\_tri}\nabla(\boldsymbol{w}_{\mathrm{m}} \cdot \boldsymbol{w}_m) \\ &= \frac{1}{2}\sum_m^{n\_tri}\nabla\boldsymbol{w}_{\mathrm{m},\nu} \cdot \boldsymbol{w}_m\end{aligned} \tag{B-22}$$

using the identity for vector calculation.

The $\nabla\boldsymbol{w}$ term is

$$\nabla\boldsymbol{w}_{m,\nu} = \left\{\nabla\boldsymbol{Q}_{\mathrm{m},1} \times \boldsymbol{Q}_{m,2} - \nabla\boldsymbol{Q}_{\mathrm{m},2} \times \boldsymbol{Q}_{m,1}\right\}, \nu = 1, 2, 3, \ldots \tag{B-23}$$

As a result,

$$\frac{\partial\sigma_1}{\partial\boldsymbol{r}_\nu} = \frac{1}{2}\sum_m^{n\_tri}\begin{bmatrix}\nabla\boldsymbol{Q}_{\mathrm{m},1}(\nu)\left(\boldsymbol{Q}_{m,1}(\boldsymbol{Q}_{m,2} \cdot \boldsymbol{Q}_{m,2}) - \boldsymbol{Q}_{m,2}(\boldsymbol{Q}_{m,1} \cdot \boldsymbol{Q}_{m,2})\right) \\ -\nabla\boldsymbol{Q}_{\mathrm{m},2}(\nu)\left(\boldsymbol{Q}_{m,1}(\boldsymbol{Q}_{m,1} \cdot \boldsymbol{Q}_{m,2}) - \boldsymbol{Q}_{m,2}(\boldsymbol{Q}_{m,1} \cdot \boldsymbol{Q}_{m,1})\right)\end{bmatrix} \tag{B-24}$$

Using Eq. B-24, we can also verify that we get Eq. B-8a through Eq. B-8c for triangular model and Eq. B-17 for tetrahedron model.

Appendix C

Calculation of Metric Matrices for One and Two Constraints

For one constraint, metric matrix is 1×1 matrix where

$$\overline{g}_{11} = \overline{G}_{11}{}^{-1} = \frac{\zeta}{\sum_{\nu=1}^{N} \nabla_\nu\sigma\cdot\nabla_\nu\sigma} \tag{C-1}$$

Therefore, the Lagrange multiplier is

$$\lambda^M = \zeta \frac{\sigma}{\sum_{\nu=1}^{N} \nabla_\nu\sigma\cdot\nabla_\nu\sigma} \tag{C-2}$$

and the iteration for the correction is

$$\boldsymbol{r}_\nu^{CON}(t+\Delta t) = \boldsymbol{r}_\nu^{UN}(t+\Delta t) - \frac{\sigma}{\sum_{\nu=1}^{N} \nabla_\nu\sigma\cdot\nabla_\nu\sigma}\nabla_\nu\sigma \tag{C-3}$$

For two constraints, metric matrix $\overline{g}$ is 2×2 matrix.

Two Lagrange multipliers are obtained.

$$\lambda_1^M = \overline{g}_{11}\sigma_1 + \overline{g}_{12}\sigma_2 \tag{C-4a}$$

$$\lambda_2^M = \overline{g}_{21}\sigma_1 + \overline{g}_{22}\sigma_2 \tag{C-4b}$$

Then,

$$\boldsymbol{r}_\nu^{CON}(t+\Delta t) = \boldsymbol{r}_\nu^{UN}(t+\Delta t) - \frac{1}{\zeta}\left[\left(\overline{g}_{11}\sigma_1 + \overline{g}_{12}\sigma_2\right)\nabla_\nu\sigma_1 + \left(\overline{g}_{21}\sigma_1 + \overline{g}_{22}\sigma_2\right)\nabla_\nu\sigma_2\right] \tag{C-5}$$

where scalar $\overline{G}_{jk}$ are

$$\overline{G}_{11} = \frac{1}{\zeta}\sum_{\nu=1}^{N} \nabla_\nu\sigma_1\cdot\nabla_\nu\sigma_1 \tag{C-6a}$$

$$\overline{G}_{12} = \frac{1}{\zeta}\sum_{\nu=1}^{N} \nabla_\nu\sigma_1\cdot\nabla_\nu\sigma_2 \tag{C-6b}$$

$$\overline{G}_{21} = \frac{1}{\zeta}\sum_{\nu=1}^{N} \nabla_\nu\sigma_2\cdot\nabla_\nu\sigma_1 \tag{C-6c}$$

$$\overline{G}_{22} = \frac{1}{\zeta}\sum_{\nu=1}^{N} \nabla_\nu\sigma_2\cdot\nabla_\nu\sigma_2 \tag{C-6d}$$

The relationship between metric matrices and modified metric matrices are

$$\overline{g}_{11} = \frac{\overline{G}_{22}}{|\overline{\boldsymbol{G}}|}, \quad \overline{g}_{12} = -\frac{\overline{G}_{12}}{|\overline{\boldsymbol{G}}|}, \quad \overline{g}_{21} = \frac{\overline{G}_{21}}{|\overline{\boldsymbol{G}}|}, \quad \overline{g}_{22} = \frac{\overline{G}_{11}}{|\overline{\boldsymbol{G}}|} \tag{C-7}$$

The determinant of a modified metric matrix is

$$det\overline{\boldsymbol{G}} = \left|\overline{\boldsymbol{G}}\right| = \overline{G}_{11}\overline{G}_{22} - \overline{G}_{12}\overline{G}_{21} \tag{C-8}$$

In Appendix D, this determinant is proved that the value is not zero and also shown that $\overline{G}_{12}$ and $\overline{G}_{21}$ is equal.

Appendix D

Proof of the value of Determinant of Modified Metric Matrix

Presented below is the proof that the determinant of modified metric matrix in the constraint subroutine is not zero. In addition, it is shown that (1,2) and (2,1) component of the modified metric matrix is equal. Following is a sample MATLAB code and the result for tetrahedron model.

```
%=========================================================
% Time-stamp: "3:29 PM 9/21/2010 415CU7 kkim32"
% This is program to check if det(G) is zero or not
% when G = SUM (DEL(sigma_old).DEL(sigma_old))
% For equalateral triangles as initial position, det(G) is practically zero
% G is modified metric matrix
% Prove det(G) = G(1,1)*G(2,2)-G(1,2)*G(2,1) /= 0
% Also, prove that G(1,2) = G(2,1)
%=========================================================
syms x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4
% position
r1 = [x1 y1 z1];
r2 = [x2 y2 z2];
r3 = [x3 y3 z3];
r4 = [x4 y4 z4];
% connector vectors
Q1 = r2 - r1;
Q2 = r3 - r1;
Q3 = r4 - r1;
Q4 = r3 - r2;
Q5 = r4 - r2;
Q6 = r4 - r3;
% cross product of connector vector
Q1XQ2 = cross(Q1,Q2);
Q2XQ3 = cross(Q2,Q3);
Q3XQ1 = cross(Q3,Q1);
Q4XQ5 = cross(Q4,Q5);
% dot product of connector vector
Q1dQ1 = sum(Q1.*Q1);
Q1dQ2 = sum(Q1.*Q2);
Q1dQ3 = sum(Q1.*Q3);
Q1dQ4 = sum(Q1.*Q4);
Q1dQ5 = sum(Q1.*Q5);
Q2dQ1 = Q1dQ2;
Q2dQ2 = sum(Q2.*Q2);
Q2dQ3 = sum(Q2.*Q3);
Q2dQ4 = sum(Q2.*Q4);
Q2dQ5 = sum(Q2.*Q5);
Q3dQ1 = Q1dQ3;
Q3dQ2 = Q2dQ3;
Q3dQ3 = sum(Q3.*Q3);
Q3dQ4 = sum(Q3.*Q4);
Q3dQ5 = sum(Q3.*Q5);
```

```
Q4dQ1 = Q1dQ4;
Q4dQ2 = Q2dQ4;
Q4dQ3 = Q3dQ4;
Q4dQ4 = sum(Q4.*Q4);
Q4dQ5 = sum(Q4.*Q5);
Q5dQ1 = Q1dQ5;
Q5dQ2 = Q2dQ5;
Q5dQ3 = Q3dQ5;
Q5dQ4 = Q4dQ5;
Q5dQ5 = sum(Q5.*Q5);
% scalar component A(v,:) where v is number of beads
A(1,1) = -Q2dQ3+Q1dQ2+Q1dQ3-Q3dQ3;
A(1,2) = Q1dQ2-Q1dQ1-Q3dQ3+Q2dQ3;
A(1,3) = Q2dQ3-Q2dQ2-Q1dQ1+Q1dQ3;
A(1,4) = 0;
A(1,5) = 0;

A(2,1) = Q2dQ2+Q3dQ3;
A(2,2) = -Q1dQ2;
A(2,3) = -Q1dQ3;
A(2,4) = -Q5dQ5+Q4dQ5;
A(2,5) = Q4dQ5-Q4dQ4;

A(3,1) = -Q1dQ2;
A(3,2) = Q1dQ1+Q3dQ3;
A(3,3) = -Q2dQ3;
A(3,4) = Q5dQ5;
A(3,5) = -Q4dQ5;

A(4,1) = -Q1dQ3;
A(4,2) = -Q2dQ3;
A(4,3) = Q2dQ2+Q1dQ1;
A(4,4) = -Q4dQ5;
A(4,5) = Q4dQ4;

param1 = 1/2;
param2 = 1/6;
%  sigma1 : AREA
%  DEL(sigma1)
dSIG1dr1 = param1*(A(1,1)*Q1+A(1,2)*Q2+A(1,3)*Q3+A(1,4)*Q4+A(1,5)*Q5);
dSIG1dr2 = param1*(A(2,1)*Q1+A(2,2)*Q2+A(2,3)*Q3+A(2,4)*Q4+A(2,5)*Q5);
dSIG1dr3 = param1*(A(3,1)*Q1+A(3,2)*Q2+A(3,3)*Q3+A(3,4)*Q4+A(3,5)*Q5);
dSIG1dr4 = param1*(A(4,1)*Q1+A(4,2)*Q2+A(4,3)*Q3+A(4,4)*Q4+A(4,5)*Q5);
%  sigma2 : VOL
%  DEL(sigma2)
dSIG2dr1 = param2 * (-Q3XQ1 - Q1XQ2 - Q2XQ3);
dSIG2dr2 = param2 * (Q2XQ3);
dSIG2dr3 = param2 * (Q3XQ1);
dSIG2dr4 = param2 * (Q1XQ2);
% modified metric matrix: G(a,b) = sum(dSIGadrv .* dSIGbdrv)
G(1,1) = sum(dSIG1dr1 .* dSIG1dr1) + sum(dSIG1dr2 .* dSIG1dr2)
       + sum(dSIG1dr3 .* dSIG1dr3) + sum(dSIG1dr4 .* dSIG1dr4);
G(1,2) = sum(dSIG1dr1 .* dSIG2dr1) + sum(dSIG1dr2 .* dSIG2dr2)
       + sum(dSIG1dr3 .* dSIG2dr3) + sum(dSIG1dr4 .* dSIG2dr4);
G(2,1) = sum(dSIG2dr1 .* dSIG1dr1) + sum(dSIG2dr2 .* dSIG1dr2)
```

```
       + sum(dSIG2dr3 .* dSIG1dr3) + sum(dSIG2dr4 .* dSIG1dr4);
G(2,2) = sum(dSIG2dr1 .* dSIG2dr1) + sum(dSIG2dr2 .* dSIG2dr2)
       + sum(dSIG2dr3 .* dSIG2dr3) + sum(dSIG2dr4 .* dSIG2dr4);
% proved that following is zero
G12MG21 = G(1,2)-G(2,1);
% [OUTCOME] G12MG21 = 0

% determinant of G
detG = G(1,1)*G(2,2)-G(1,2)*G(2,1);
% metric matrix: g = inv(G)
g(1,1) = G(2,2)/detG;
g(1,2) = -G(1,2)/detG;
g(2,1) = -G(2,1)/detG;
g(2,2) = G(1,1)/detG;
```

Appendix E

Velocity Field and Stress Tensor for Different Flow Types

Homogeneous velocity fields with $\boldsymbol{v}_0 = 0$ in Cartesian coordinate is

$$\boldsymbol{v}(v_x, v_y, v_z) \tag{E-1}$$

Where each component is written explicitly as

$$v_x = \kappa_{xx}x + \kappa_{xy}y + \kappa_{xz}z \tag{E-2a}$$

$$v_y = \kappa_{yx}x + \kappa_{yy}y + \kappa_{yz}z \tag{E-2b}$$

$$v_z = \kappa_{zx}x + \kappa_{zy}y + \kappa_{zz}z \tag{E-2c}$$

The incompressible flow can be expressed as the first invariant to be zero.

$$\text{I} = tr(\boldsymbol{\kappa}) = \sum_i \kappa_{ii} = 0 \text{ (traceless) or } \nabla \cdot v = 0 \tag{E-3}$$

Shear flow

The $\boldsymbol{\kappa}$ in the shear flow is a rate-of-strain tensor.

$$\kappa_{xy} = \dot{\gamma} \tag{E-4a}$$

$$\boldsymbol{\kappa} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \dot{\gamma} = (\nabla \boldsymbol{v})^{\dagger} \tag{E-4b}$$

$$\dot{\boldsymbol{\gamma}} = \nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^{\dagger} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \frac{\partial v_x}{\partial y} \tag{E-4c}$$

The superscript † indicates the transpose of the tensor.

The invariants of $\dot{\boldsymbol{\gamma}}$ are calculated below.

$$\text{First invariant I} = tr(\dot{\boldsymbol{\gamma}}) = 2\sum_{\text{i}} \frac{\partial v_i}{\partial x_i} = 2\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}\right) = 2\nabla \cdot \boldsymbol{v} = 0 \tag{E-5a}$$

$$\text{Second invariant II} = tr(\dot{\boldsymbol{\gamma}}^2) = tr\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \dot{\gamma}_{yx}^2 \tag{E-5b}$$

$$\text{Third invariant III} = tr(\dot{\boldsymbol{\gamma}}^3) = tr\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\left(\frac{\partial v_x}{\partial y}\right)^3 = 0 \tag{E-5c}$$

The shear rate $\dot{\gamma}$ is a scalar that is relate to the second invariant of rate-of-strain tensor.

$$\dot{\gamma} = \sqrt{\frac{1}{2}\dot{\boldsymbol{\gamma}}:\dot{\boldsymbol{\gamma}}} \tag{E-6}$$

The homogeneous Giesekus form of tensor in Eq. 3.39 (Section 3.4) is then

$$\boldsymbol{\tau}_c = 2\sum_\nu \left\{\frac{d}{dt}\langle \boldsymbol{R}_\nu \boldsymbol{R}_\nu\rangle - \boldsymbol{\kappa}\cdot\langle \boldsymbol{R}_\nu \boldsymbol{R}_\nu\rangle - \langle \boldsymbol{R}_\nu \boldsymbol{R}_\nu\rangle\cdot\boldsymbol{\kappa}^\dagger\right\} \tag{E-7}$$

after substituting $n_c H = 1$ and $\zeta = 4H$ (Bird 1987).

Then components of the stress tensor we need to calculate the material functions are

$$\begin{aligned}\tau_{xy} = \tau_{yx} &= 2\left\{\frac{d}{dt}\sum_\nu\langle \boldsymbol{R}_1\boldsymbol{R}_2\rangle - \dot{\gamma}\sum_\nu\langle \boldsymbol{R}_2\boldsymbol{R}_2\rangle\right\}\\ &= 2\frac{d}{dt}\sum_\nu\left(r_{\nu,x}-r_{c,x}\right)\left(r_{\nu,y}-r_{c,y}\right) - 2\dot{\gamma}\sum_\nu\left(\left(r_{\nu,y}-r_{c,y}\right)^2\right)\end{aligned} \tag{E-8}$$

$$\begin{aligned}\tau_{xx} &= 2\left\{\frac{d}{dt}\sum_\nu\langle \boldsymbol{R}_1\boldsymbol{R}_1\rangle - 2\dot{\gamma}\sum_\nu\langle \boldsymbol{R}_1\boldsymbol{R}_2\rangle\right\}\\ &= 2\frac{d}{dt}\sum_\nu\left(\left(r_{\nu,x}-r_{c,x}\right)^2\right) - 4\dot{\gamma}\sum_\nu\left\{\left(r_{\nu,x}-r_{c,x}\right)\left(r_{\nu,y}-r_{c,y}\right)\right\}\end{aligned} \tag{E-9}$$

$$\begin{aligned}\tau_{yy} &= 2\left\{\frac{d}{dt}\sum_\nu\langle \boldsymbol{R}_2\boldsymbol{R}_2\rangle\right\}\\ &= 2\frac{d}{dt}\sum_\nu\left(\left(r_{\nu,y}-r_{c,y}\right)^2\right)\end{aligned} \tag{E-10}$$

where $r_{\nu,x}$ is the x-coordinate of the position of bead $\nu$and $r_{c,x}$ is that of center-of-mass.

Stress Relaxation after Sudden Shearing Displacement

The $\boldsymbol{\kappa}$ in the shear flow is a rate-of-strain tensor.

$$\kappa_{xy} = \gamma_0\delta(t) \tag{E-11a}$$

$$\boldsymbol{\kappa} = \begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\gamma_0\delta(t) \tag{E-11b}$$

Components of the stress tensor in Eq. E-7is then

$$\tau_{xy} = \tau_{yx} = 2\left\{\frac{d}{dt}\sum_v \langle \boldsymbol{R}_1\boldsymbol{R}_2\rangle - \gamma_0\delta(t)\sum_v \langle \boldsymbol{R}_2\boldsymbol{R}_2\rangle\right\} \quad \text{(E-12)}$$

$$\tau_{xx} = 2\left\{\frac{d}{dt}\sum_v \langle \boldsymbol{R}_1\boldsymbol{R}_1\rangle - 2\gamma_0\delta(t)\sum_v \langle \boldsymbol{R}_1\boldsymbol{R}_2\rangle\right\} \quad \text{(E-13)}$$

$$\tau_{yy} = 2\left\{\frac{d}{dt}\sum_v \langle \boldsymbol{R}_2\boldsymbol{R}_2\rangle\right\} \quad \text{(E-14)}$$

Shear free flow

The $\boldsymbol{\kappa}$ in the shear free flow is elongational rate. The zz-component of $\boldsymbol{\kappa}$ is

$$\kappa_{zz} = \dot{\varepsilon} \quad \text{(E-15)}$$

The components of velocity field are

$$v_x = -\frac{1}{2}\kappa_{zz}(1+b)x \quad \text{(E-16)}$$

$$v_y = -\frac{1}{2}\kappa_{zz}(1-b)y$$

$$v_z = \kappa_{zz}z$$

where the parameter b that defines the type of flow has range of $0 \leq b \leq 1$. The elongational flow ($b = 0, \dot{\varepsilon} > 0$) gives the flow characteristic by stretching the fluid in the z-axis whereas the biaxial stretching flow ($b = 0, \dot{\varepsilon} < 0$) stretches in the direction of x-axis and y-axis. When $b = 1$, the flow is called planer elongational and there is no stretching in the y direction. The illustrations of deformation of these three shear free flows are in the volume 1 of Bird *et al.* (1987).

In the case elongational flow where $b = 0$ and $\dot{\varepsilon} > 0$, the velocity field becomes

$$v_x = -\frac{1}{2}\dot{\varepsilon}x \quad \text{(E-17a)}$$

$$v_y = -\frac{1}{2}\dot{\varepsilon}y \quad \text{(E-17b)}$$

$$v_z = +\dot{\varepsilon}z \quad \text{(E-17c)}$$

The gradient of velocity is then

$$\nabla \boldsymbol{v} = \begin{pmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_y}{\partial x} & \frac{\partial v_z}{\partial x} \\ \frac{\partial v_x}{\partial y} & \frac{\partial v_y}{\partial y} & \frac{\partial v_z}{\partial y} \\ \frac{\partial v_x}{\partial z} & \frac{\partial v_y}{\partial z} & \frac{\partial v_z}{\partial z} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 2 \end{pmatrix}\dot{\varepsilon} = (\nabla \boldsymbol{v})^{\dagger} \quad \text{(E-18)}$$

$$\boldsymbol{\kappa}^{\dagger} = \boldsymbol{\kappa} = \frac{1}{2}\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 2 \end{pmatrix}\dot{\varepsilon} \quad \text{(E-19)}$$

The invariants of $\boldsymbol{\kappa}$ are

$$\text{First invariant I} = tr(\boldsymbol{\kappa}) = 2\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}\right) = 2\left(-\frac{\dot{\varepsilon}}{2} - \frac{\dot{\varepsilon}}{2} + \dot{\varepsilon}\right) = 0 \quad \text{(E-20a)}$$

$$\text{Second invariant II} = tr(\boldsymbol{\kappa}^2) = tr\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix}\dot{\varepsilon}^2 = 6\dot{\varepsilon}^2 \quad \text{(E-20b)}$$

$$\text{Third invariant III} = tr(\boldsymbol{\kappa}^3) = tr\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 8 \end{pmatrix}\dot{\varepsilon}^3 = 6\dot{\varepsilon}^3 \quad \text{(E-20c)}$$

Note that the third invariant for shear free flow is not zero where shear flow is zero.

Similarly as calculated for the shear flow, stress tensor is

$$\boldsymbol{\tau}_c = \sum_{\nu}\left\{2\frac{d}{dt}\begin{pmatrix} \boldsymbol{R}_1\boldsymbol{R}_1 & \boldsymbol{R}_1\boldsymbol{R}_2 & \boldsymbol{R}_1\boldsymbol{R}_3 \\ \boldsymbol{R}_2\boldsymbol{R}_1 & \boldsymbol{R}_2\boldsymbol{R}_2 & \boldsymbol{R}_2\boldsymbol{R}_3 \\ \boldsymbol{R}_3\boldsymbol{R}_1 & \boldsymbol{R}_3\boldsymbol{R}_2 & \boldsymbol{R}_3\boldsymbol{R}_3 \end{pmatrix} - 2\begin{pmatrix} -\boldsymbol{R}_1\boldsymbol{R}_1 & -\boldsymbol{R}_1\boldsymbol{R}_2 & \frac{1}{2}\boldsymbol{R}_1\boldsymbol{R}_3 \\ -\boldsymbol{R}_1\boldsymbol{R}_1 & -\boldsymbol{R}_2\boldsymbol{R}_2 & \frac{1}{2}\boldsymbol{R}_2\boldsymbol{R}_3 \\ \frac{1}{2}\boldsymbol{R}_3\boldsymbol{R}_1 & \frac{1}{2}\boldsymbol{R}_3\boldsymbol{R}_2 & 2\boldsymbol{R}_3\boldsymbol{R}_3 \end{pmatrix}\dot{\varepsilon}\right\} \quad \text{(E-21)}$$

Then three components of the stress tensor we need to calculate the material functions are as follows.

$$\tau_{xx} = 2\left\{\frac{d}{dt}\sum_{\nu}\langle \boldsymbol{R}_1\boldsymbol{R}_1\rangle + \dot{\varepsilon}\sum_{\nu}\langle \boldsymbol{R}_1\boldsymbol{R}_1\rangle\right\} \quad \text{(E-22)}$$

$$= 2\frac{d}{dt}\sum_{\nu}\left\{\left(r_{\nu,x} - r_{c,x}\right)^2\right\} + 2\dot{\varepsilon}\sum_{\nu}\left\{\left(r_{\nu,x} - r_{c,x}\right)^2\right\}$$

$$\tau_{yy} = 2\left\{\frac{d}{dt}\sum_{\nu}\langle \boldsymbol{R}_2\boldsymbol{R}_2\rangle + \dot{\varepsilon}\sum_{\nu}\langle \boldsymbol{R}_2\boldsymbol{R}_2\rangle\right\} \quad \text{(E-23)}$$

$$= 2\frac{d}{dt}\sum_{\nu}\left\{\left(r_{\nu,y} - r_{c,y}\right)^2\right\} + 2\dot{\varepsilon}\sum_{\nu}\left\{\left(r_{\nu,y} - r_{c,y}\right)^2\right\}$$

$$\tau_{zz} = 2\left\{\frac{d}{dt}\sum_{\nu}\langle \boldsymbol{R}_3\boldsymbol{R}_3\rangle - 2\dot{\varepsilon}\sum_{\nu}\langle \boldsymbol{R}_3\boldsymbol{R}_3\rangle\right\} \tag{E-24}$$

$$= 2\frac{d}{dt}\sum_{\nu}\left\{\left(r_{\nu,z} - r_{c,z}\right)^2\right\} - 4\dot{\varepsilon}\sum_{\nu}\left\{\left(r_{\nu,z} - r_{c,z}\right)^2\right\}$$

The rest of the components have been proven to be zero for incompressible fluids.

Appendix F

MATLAB Code for the Initial Position of the Biconcave Model

Modeling of RBC using mathematical expression from Kuchel (1999) in Cartesian coordinate.

```
% [p,t] = distmeshsurface(@fd,@fh,h0,bbox);
% [p,t] = distmesh3d(@fd,@fh,h0,bbox,pfix,varargin);
% OUTPUT:
% p       Contains x,y,z coordinates for each of the N nodes. (Nx3) array
% t       Contains the indices of each triangle (NTx3)
% INPUT :
% fd      Geometry given as a distance function d(x,y)
% fh      Scaled edge length function.
%         Returns h for all input points h(x,y).
%         Constant for uniform meshes.
% h0      Initial edge length.
%         Distance between points in the initial distribution p0
% bbox    Bounding box for the region [xmin,ymin; xmax,ymax]
% pfix    Fixed node positions given as an array
% varargin    Additional parameters to the functions fd and fh

h0 = 1.16;
bbox = [-100,-100,-30;100,100,30];
% For biconcave RBC,
[p,t] = distmeshsurface(@discocyte,@huniform,h0,bbox);
%[p,t] = distmesh3d(@discocyte,@huniform,h0,bbox,[]);
%[p,t] = distmesh3d(@dmatrix3d,@huniform,h0,bbox,x,y,z,phi,hh);
%[p,t] = distmeshnd(@fdrbc,@fhrbc,h0,box,[]);
% For sphere,
%[p,t]=distmeshnd(@fdsph,@huniform,0.2,[-ones(1,dim);ones(1,dim)],[]);
```

The scalar value of $h0$ gives the distance between beads. The program will run until the value of h0 is optimized and yet satisfies the surface shape of the RBC to get the initial distribution of the beads. The following is the MATLAB function showing the mathematical expression of Kuchel (1999) to model RBC surface in Cartesian coordinate that is used to construct the healthy normal RBC. Diameter and thickness of RBC is scaled to have 10 dimensionless unit.

```
function phi=discocyte(p)
% For a mature normal erythrocyte, biconcave disk without a nucleus
r2 = p(:,1).^2 + p(:,2).^2;
z = p(:,3);
```

```
% d                                       : Diameter of the RBC.
d = 7.8*1.4;                              % (7.8 micron) Martin 2006
% 2a                                      : Thickness of the RBC at the center
a = 0.5*1.4;                              % (0.5 micron) Martin 2006
% m [0,1] ; controls the maximum thickness of the cell
m = 0.9447;                               % Martin 2006
% V = 85.1                                % (micron^3) volume Martin 2006
% See paper for the expression for V
P = (1 - 2*m)*d.^2 / (4*m);
Q = (1 - m)*d.^4 / (16*a.^2*m);
R = (m - 1)*d.^4 / (16*m);
%                                         % Larkin and Kuchel 2010
% Phi(x,y,z) = (x2 + y2)2 + P(x2 + y2) + Qz2 + R
% phi = r2.^2 + P*r2 + Q*z.^2 + R;
%                                         % Kuchel and Fackerell 1999
% Phi(x,y,z) = (x2 + y2 + z2)2 + P(x2 + y2) + Qz2 + R
phi = (r2 + p(:,3).^2).^2 + P * r2 + Q * z.^2 + R;


% For a sphere geometry given as a distance function
function d=fdsph(p)
d=sqrt(sum(p.^2,2))-1;
% sum(p.^2,2) = p(:,1).^2+p(:,2).^2+p(:,3).^2
dsphere = 0.65-sqrt(p(:,1).^2+p(:,2).^2+p(:,3).^2);
d=max(d,dsphere);
```

Delaunay triangulation algorithm using distance functions by Persson (2005) is used to obtain the initial position of biconcave RBC and all the indices of the bead positions. A sample code of 3D surface meshing algorithm is provided below. This algorithm is not limited to biconcave shape (e.g. function discocyte) and can apply to uniform size functions. Note that not all the functions and routine calls are listed in this appendix.

```
% DISTMESHSURFACE 3-D Surface Mesh Generator using Distance Functions
% by Per-Olof Persson (2005)
function [p,t]=distmeshsurface(fd,fh,h0,bbox,varargin)

% kkim32: Save the result figure as an animation file
vidObj = VideoWriter('RBCanimation.avi');
vidObj.Quality = 100;
vidObj.FrameRate= 3;
open(vidObj);

dptol=1e-4; ttol=.1; Fscale=1.2; deltat=.2; deps=sqrt(eps)*h0;

% 1.    Create initial distribution in bounding box (isosurface from grid)
[x,y,z]=ndgrid(bbox(1,1):h0:bbox(2,1),bbox(1,2):h0:bbox(2,2),bbox(1,3):h0:b
```

```
box(2,3));
pv=isosurface(x,y,z,reshape(fd([x(:),y(:),z(:)],varargin{:}),size(x)),0);
p=pv.vertices;
t=pv.faces;
% 2.   Connectivities (for trisurfupd)
[t2t,t2n]=mkt2t(t);
t2t=int32(t2t-1)'; t2n=int8(t2n-1)';
N=size(p,1);                                        % Number of points N
pold=inf;                                           % For first iteration
while 1
  p0=p;
  % 3.    Retriangulation
  if max(sqrt(sum((p-pold).^2,2))/h0)>ttol          % Any large movement?
    pold=p;                                         % Save current positions
    [t,t2t,t2n]=trisurfupd(int32(t-1)',t2t,t2n,p');% Update triangles
    t=double(t+1)';
    pmid=(p(t(:,1),:)+p(t(:,2),:)+p(t(:,3),:))/3;  % Compute centroids
    % 4.    Describe each bar by a unique pair of nodes
    bars=[t(:,[1,2]);t(:,[1,3]);t(:,[2,3])];     % Interior bars duplicated
    bar=sort(bars,2);
    % barr=bar(521:530,[1,2])
    bars=unique(sort(bars,2),'rows');               % Bars as node pairs
    % 5.    Graphical output of the current mesh
    %clf,patch('faces',t,'vertices',p,'facecol',[1,0,0],'edgecol','k');

    % <-- kkim32
    % Plot of the positions
    % simpplot(p,t,'p(:,1)<0');                     % plot half
    simpplot(p,t);                                  % plot all
    f = getframe;
    writeVideo(vidObj,f);
    % Labeling the nodes
    m=[p(t(:),1),p(t(:),2),p(t(:),3)];
    mu=unique(m,'rows');
    %th=text(mu(:,1),mu(:,2),mu(:,3),num2cell(1:size(mu,1)),'fontsize',8);
    th=text(mu(1:15,1),mu(1:15,2),mu(1:15,3),num2cell(1:15),'fontsize',15);
    % -->

    %axis equal;axis off;view(3);cameramenu;drawnow
  end

  % 6.    Move mesh points based on bar lengths L and forces F
  barvec=p(bars(:,1),:)-p(bars(:,2),:);             % List of bar vectors
  L=sqrt(sum(barvec.^2,2));                         % L = Bar lengths
  hbars=feval(fh,(p(bars(:,1),:)+p(bars(:,2),:))/2,varargin{:});
  L0=hbars*Fscale*sqrt(sum(L.^2)/sum(hbars.^2));   % L0 = Desired lengths
  F=max(L0-L,0);                                    % Bar forces (scalars)
  Fvec=F./L*[1,1,1].*barvec;                % Bar forces (x,y,z components)

Ftot=full(sparse(bars(:,[1,1,1,2,2,2]),ones(size(F))*[1,2,3,1,2,3],[Fvec,-
Fvec],N,3));
  p=p+deltat*Ftot;                                  % Update node positions

  % 7.    Bring all points back to the boundary
  d=feval(fd,p,varargin{:});
```

```
  %                                                    % Numerical
  dgradx=(feval(fd,[p(:,1)+deps,p(:,2),p(:,3)],varargin{:})-d)/deps;
  %                                                    % gradient
  dgrady=(feval(fd,[p(:,1),p(:,2)+deps,p(:,3)],varargin{:})-d)/deps;
  dgradz=(feval(fd,[p(:,1),p(:,2),p(:,3)+deps],varargin{:})-d)/deps;
  dgrad2=dgradx.^2+dgrady.^2+dgradz.^2;
  %                                                  % Project back to boundary
  p=p-[d.*dgradx./dgrad2,d.*dgrady./dgrad2,d.*dgradz./dgrad2];

  % 8.    Termination criterion: All nodes move less than dptol (scaled)
  if max(sqrt(sum((p-p0).^2,2))/h0)<dptol, break; end
end

close(vidObj);
winopen('animation.avi')
```

Appendix G

Selected Tabulated Material Functions Data

TABLE VIII
STEADY STATE VISCOSITY DATA:
TWO-CONSTRAINT TRIANGLE MODEL
$a = \sum(\text{area})^2$ =100, N[a] = 100000

Shear rate	Viscosity		First Normal Stress Coefficient	
	Mean	SD[b]	Mean	SD[b]
0.03	17.49895	0.10554	78.22529	6.05346
0.05	17.21468	0.10480	82.93406	3.61387
0.08	16.44187	0.10354	70.43676	2.18357
0.14	15.09160	0.10124	57.76792	1.29798
0.30	12.38448	0.09292	35.05726	0.53434
0.50	10.35347	0.08215	23.29330	0.29981
0.82	8.57062	0.07108	14.89336	0.16431
1.35	6.95398	0.05966	8.73350	0.08963
2.23	6.00090	0.05670	5.11169	0.04962
3.67	5.31736	0.05553	2.98024	0.02770
6.05	4.78847	0.05465	1.69725	0.01545
9.97	4.17397	0.05304	0.95872	0.00877
16.44	3.74467	0.05106	0.52778	0.00499
27.11	3.25934	0.04908	0.29468	0.00284
44.70	2.85535	0.04653	0.16333	0.00161
73.70	2.50128	0.04542	0.08801	0.00094
90.02	2.45235	0.04454	0.07055	0.00076

[a] number of trajectories.
[b] standard deviation.

TABLE IX
SUDDEN DISPLACEMENT DATA:
TWO-CONSTRAINT TRIANGLE MODEL
$a = \sum(\text{area})^2$ =100, N[a] = 100000

Time	G modulus		Gpsi modulus	
	Mean	SD[b]	Mean	SD[b]
0.0004	1675.98001	10.27759	0.44237	0.00841
0.0006	554.57244	7.97822	0.17233	0.00739
0.0008	331.67137	6.26625	0.09898	0.00656
0.001	194.45828	5.25403	0.05467	0.00608
0.002	63.57219	1.90032	0.02649	0.00252
0.004	20.97707	1.10628	0.01359	0.00168
0.006	15.22199	1.05165	0.01251	0.00168
0.008	13.05975	1.03696	0.01446	0.00167
0.010	11.89818	1.01423	0.01255	0.00167
0.020	7.43251	0.43798	0.01056	0.00075
0.040	3.10912	0.29602	0.00935	0.00053
0.060	1.21032	0.29163	0.00909	0.00053
0.080	1.72320	0.29103	0.00869	0.00053
0.100	0.87101	0.29070	0.00810	0.00053
0.200	1.80519	0.12980	0.00877	0.00023
0.400	1.58106	0.09144	0.00841	0.00017
0.600	1.44305	0.09155	0.00832	0.00017
0.800	1.41885	0.09152	0.00772	0.00017
1.000	1.39459	0.09156	0.00725	0.00007
2.000	1.34548	0.03997	0.00631	0.00007
3.000	1.16469	0.04005	0.06923	0.00105

[a] number of trajectories.
[b] standard deviation.

TABLE X
ELONGATIONAL VISCOSITY DATA:
TWO-CONSTRAINT TETAHEDRON MODEL
$a' = \sum(\text{area})^2$ =100, N[a] = 100000

Elongational rate	Elongational Viscosity		Gyration	
	Mean	SD[b]	Mean	SD[b]
54.59815	66.86314	0.00134	2.30273	0.00001
33.11545	66.80118	0.00172	2.30249	0.00001
20.08554	66.67966	0.00247	2.30210	0.00001
12.18249	66.41815	0.00403	2.30144	0.00002
7.38906	65.85665	0.00700	2.30040	0.00003
4.48169	64.57229	0.01328	2.29854	0.00004
2.71828	61.70145	0.02703	2.29496	0.00008
1.64872	56.26433	0.05648	2.28480	0.00022
1.00000	50.19397	0.08069	2.26627	0.00036
0.60653	45.84464	0.08710	2.25049	0.00042
0.36788	43.47376	0.08619	2.24228	0.00044
0.22313	41.92541	0.08420	2.23727	0.00044
0.13534	41.07066	0.08320	2.23491	0.00044
0.08208	41.07066	0.08167	2.23436	0.00044
0.04979	40.27892	0.08109	2.23363	0.00045
0.03020	40.10661	0.08061	2.23346	0.00045
0.01832	40.06649	0.08191	2.23285	0.00045
0.01111	40.08654	0.08065	2.23217	0.00045
0.00674	39.91831	0.08068	2.23318	0.00044
0.00409	39.91433	0.08091	2.23271	0.00044
0.00248	39.80886	0.08144	2.23247	0.00045
0.00150	39.99004	0.08111	2.23318	0.00044
0.00091	40.00605	0.08072	2.23331	0.00044
0.00055	39.96790	0.08125	2.23278	0.00044
0.00034	39.84075	0.07989	2.23237	0.00044
0.00020	39.89897	0.08093	2.23270	0.00044

[a] number of trajectories.
[b] standard deviation.
Computation time: 2074630.5 (sec)

Appendix H
Fortran Code for the Simulation

This appendix contains the list of modules, functions, and subroutines that are used in the program along with a sample Fortran code. The number of trajectories (n_tra) range between $10^4 \sim 10^6$ in which the material properties are averaged. The range of shear rate is between $0.01 \sim 1000$ dimensionless unit. The maximum number of iterations (maxiter) is set to 100 in the constraint subroutine. The Lagrange multiplier convergence tolerance (tol) to satisfy each constraint is set to 1.5×10^{-3}. These values of parameters were chosen to make the simulation converge with low standard deviation. The rest of the parameters are shown in Table V in Section 4.

In the simulation code, the derived data types in Fortran is used to contain coordinate information in one variable (or object). In addition, declaration of custom operators shown in the module vec_func makes the form of vector calculations much simpler and easy to read. This will allow us to generalize the form of equations used in the code so that we can systematically increase the number of beads in the microstructure. For example, we can use the custom cross product operator to calculate cross product of two vectors $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ by

$$\boldsymbol{w}_1 = \boldsymbol{Q}_1 .\text{X}. \boldsymbol{Q}_2$$

Each vector contains subcomponent of x, y, and z coordinates as follow.

$$\begin{cases} \boldsymbol{Q}_1\%x \\ \boldsymbol{Q}_1\%y \\ \boldsymbol{Q}_1\%z \end{cases}, \begin{cases} \boldsymbol{Q}_2\%x \\ \boldsymbol{Q}_2\%y \\ \boldsymbol{Q}_2\%z \end{cases}$$

The operator is linked to function crossVEC so that each x, y, and z component of $\boldsymbol{w}_1$ is calculated and returns as a package.

```
!%   Copyright (C) 2009-2015 Kyung Hyo Kim
! ------------------------------------------------------------------------
!    [MODULES]
!    module Numeric_kinds         : Define variable types
!    module constants             : Constants and logicals
!    module vec_func              : Vector functions
!       contains
!       function absVEC           : Absolute of a vector
!       function dotVEC           : Dot product of two vectors
!       function crossVEC         : Cross product of two vectors
!       function addVEC           : Addition of two vectors
!       function subtVEC          : Subtraction of two vectors
!       function multiVEC         : Multiplication of scalar and vector
!    module Global                : Globally used parameters
!    module info                  : Contains file information
!    module RNGcommon             : Constants and common variables
!    for Random Number Generator
!
!    program RBCHOOKE             : Main program
!
!    [TYPE OF FLOW]
!    subroutine FLOW              : Shear/elongational flow
!    subroutine NoFLOW            : No flow condition (no shear)
!    subroutine BOUNDARY_FLOW     : Capillary flow
!
!    [SUBROUTINES]
!    subroutine t_setup           : Array of time
!    subroutine srt_setup         : Array of shear rate
!    subroutine srt_test          : deltat for selected shear rate
!    subroutine init_position     : Initial position of the beads
!    subroutine constrnt_area     : Calculation of area constraint
!    subroutine constrnt_len      : Total length constraint for triangular
!    subroutine constrnt_vol      : Total volume constraint for tetrahedron
!    subroutine constrnt_com_no   : Calculation of two constraints
!                                   DEL(SIGnew).DEL(SIGold)
!    subroutine constrnt_com_oo   : Calculation of two constraints
!                                   DEL(SIGold).DEL(SIGold)
!    subroutine center_mass       : Calculation of center-of-mass
!    subroutine R_position        : Calculate distance from the
!                                   center of mass
!    subroutine connector         : Calculation of connector vectors
!    subroutine UNIQUEconnector   : Unique connector vectors from
!                                   sets of connector vectors
!    subroutine SORTpairnodes     : sort the row of each pair nodes
!    subroutine InterParticleForces : Inter-particle Forces
!                                      for each points
!    subroutine crossProductQ1Q2  : scalar component of ds1point
!                                   (area constraint) for triangles
!    subroutine fit               : Linear extrapolation (regression)
!    subroutine ranils            : Initializes random number generators
!
!    [FUNCTIONS]
!    function area2               : Calculate sum of (area)^2 of a triangle
!    function ranuls              : Uniform Random Number Generator
!    function rangls              : Gaussian Random Number Generator
```

```
!    function ranils              : Subroutine to initiate ranuls
! ------------------------------------------------------------------------
!@ module vec_func - vector functions
   module vec_func
      use Numeric_kinds
      implicit none
      type coord
!     x,y,z                       : Refers to x,y,z coordinate of bead
      real(R8K)                   :: x,y,z
      end type coord
      interface abs
         module procedure absVEC
      end interface
      interface operator(.dot.)
         module procedure dotVEC
      end interface
      interface operator(.X.)
         module procedure crossVEC
      end interface
      interface operator(.A.)
         module procedure addVEC
      end interface
      interface operator(.S.)
         module procedure subtVEC
      end interface
      interface operator(.M.)
         module procedure multiVEC
      end interface
      contains
! ------------------------------------------------------------------------
!     a                     Vector
!     b                     Vector
!     s                     Scalar
!     x,y,z                 Refers to x,y,z coordinate of bead
!     absVEC                Absolute of vector
!     dotVEC                Dot product of two vectors
!     crossVEC              Cross product of two vectors
!     addVEC                Addition of two vectors (element by element)
!     subtVEC               Subtraction of two vectors (element by element)
!     multiVEC              Multiplication of scalar and vector
! ------------------------------------------------------------------------
!@ Absolute of vector
      function absVEC(a)
         type(coord), intent(IN)       :: a
         real(R8K)                     :: absVEC
         absVEC = SQRT(a%x**2 + a%y**2 + a%z**2)
      end function absVEC
! ------------------------------------------------------------------------
!@ Dot product of two vectors
      function dotVEC(a,b)
         type(coord), intent(IN)       :: a,b
         real(R8K)                     :: dotVEC
         dotVEC = a%x*b%x + a%y*b%y + a%z*b%z
      end function dotVEC
! ------------------------------------------------------------------------
```

```
!@ Cross product of two vectors
      function crossVEC(a,b)
         type(coord), intent(IN)       :: a,b
         type(coord)                   :: crossVEC
         crossVEC%x = a%y*b%z - a%z*b%y
         crossVEC%y = a%z*b%x - a%x*b%z
         crossVEC%z = a%x*b%y - a%y*b%x
      end function crossVEC
! ------------------------------------------------------------------------
!@ The addition operator implements element-by-element addition between two
vectors
      function addVEC(a,b)
         type(coord), intent(IN)       :: a,b
         type(coord)                   :: addVEC
         addVEC%x = a%x + b%x
         addVEC%y = a%y + b%y
         addVEC%z = a%z + b%z
      end function addVEC
! ------------------------------------------------------------------------
!@  The  subtraction  operator  implements  element-by-element  subtraction
between two vectors
      function subtVEC(a,b)
         type(coord), intent(IN)       :: a,b
         type(coord)                   :: subtVEC
         subtVEC%x = a%x - b%x
         subtVEC%y = a%y - b%y
         subtVEC%z = a%z - b%z
      end function subtVEC
! ------------------------------------------------------------------------
!@ function multiVEC - The multiplication operator implements
!@                    element-by-element multiplication scalar with vector
      function multiVEC(s,a)
         real(R8K), intent(IN)         :: s
         type(coord), intent(IN)       :: a
         type(coord)                   :: multiVEC
         multiVEC%x = s * a%x
         multiVEC%y = s * a%y
         multiVEC%z = s * a%z
      end function multiVEC

end module vec_func
! ------------------------------------------------------------------------
!@ module Global - globally used parameters

   module Global

      use constants
      use Numeric_kinds
      use vec_func
!     ----------------------------------------------------------------
!     Constants used in this code:
!     ----------------------------------------------------------------
!     n_bead            Number of beads
!     n_tri             Number of triangles in RBC model
!     n_data            Number of different deltats for extrapolation
```

```
!     n_srt             Number of shear rate points
!     n_elong           Number of set of elongational rate points
!     n_time            Number of time steps
!     n_t               Number of time points
!     n_tra             Number of trajectories
!     tmax              Total dimensionless time
!     deltat            Time step size
!     time_equil        Dimensionless time to meet equilibrium initial
!                       condition
!     maxiter           Maximum iteration in the constraint subroutine
!     tol               Tolerence to converge
!     tol_slope         Tolerence to determine the st. st.
!     minus             Negative sign for vector calculaton
!     TriIndices        Indices of each triangle (fixed values)
!     ------------------------------------------------------------
!     Variables used in this code:
!     ------------------------------------------------------------
!     xEta              Set of deltats
!     yEta              Set of viscosity at different deltats
!     ydEta             Standard deviation of yEta
!     EXTRA_Eta         Extrapolated viscosity to deltat=0
!     EXTRA_dEta        Standard deviation of EXTRA_Eta
!     Bead              Position of beads
!     NewBead           New position of beads
!     Q                 Connector vectors
! ------------------------------------------------------------------------
!@ program RBCHOOKE - Biconcave Model
   Program RBCHOOKE
      use Global
      implicit none

!     ------------------------------------------------------------
!     Flow type
!     ------------------------------------------------------------
!     SHEAR           Shear flow
!     SHEARFREE       Shear-free flow
!     CAPILLARY       Capillary flow
!     ------------------------------------------------------------
!     Flow condition
!     ------------------------------------------------------------
!     STUP            Start-Up
!     STST            Steady-State
!     DISP            Sudden Shearing Displacement: Relaxation experiment
!     ------------------------------------------------------------
!     Constraints
!     ------------------------------------------------------------
!     AREA            Area constraint
!     LEN             Perimeter constraint
!     VOL             Total volume constraint
!     COM_no          Combined constraint using old and new position
!                     for DEL(SIGnew).DEL(SIGold)
!     COM_oo          Combined constraint using only old position
!                     for DEL(SIGold).DEL(SIGold)
!     ------------------------------------------------------------
```

```
      flow_type     = 'SHEAR' !'SHEARFREE''CAPILLARY'
      flow_cond     = 'STST'!'DISP'!'STUP'
      areasize      = 'SUMarea2_100'  ! subroutine init_position
      constraint    = 'COM_no'!'COM_oo'!'VOL'!'AREA'

      if (constraint == 'VOL' .and. n_bead == 3) constraint    = 'LEN'

      if (flow_type == 'SHEAR')     n_rate  = n_srt
         if (flow_cond == 'DISP')     n_rate  = 1
      if (flow_type == 'SHEARFREE') n_rate  = n_elong
      if (flow_type == 'CAPILLARY') n_rate  = n_srt

      ! if STUP flow simmulation was done separately,
        then call tmax_fromSTUP
      ! the valuse was collected from the log file from STUP call.
      !if (flow_cond == 'STST') call tmax_fromSTUP()
      if (flow_type == 'CAPILLARY') then
         call CAPILLARY_FLOW()
      else
         call FLOW()

      end if

      ! Inital position of beads read from files:
      ! 1. a file containing Cartesian position of each bead
      ! 2. a file containing 3 node indices of each triangle

   end Program RBCHOOKE
! ------------------------------------------------------------------------
!@ subroutine FLOW - [Start-up/Steady-state] [Shear Flow/Shear-free Flow]
   subroutine FLOW()
! ------------------------------------------------------------------------
!     The variables used in this code are:
! ------------------------------------------------------------------------
!     sr                Shear rate
!     time              Dimensionless time
!     sq21dt            0.707*(sqrt(12.0D0*deltat))
!     0.707             SQRT(2kT/friction coeff.)
!     sqrt(12*deltat)   Part of the distribution sqrt(12dt)*(Y-0.5)
!                       where Y is uniform distribution [0,1]=ranuls
!     slope             Slope of viscosity wrt time
! ------------------------------------------------------------------------
!     Material properties:
! ------------------------------------------------------------------------
!     q21               yx component of stress tensor
!     q1122             xx-yy component of stress tensor
!     q3311             zz-xx component of stress tensor
!     aeta              Viscosity
!     apsi1             First normal stress coefficient
!     apsi2             Second normal stress coefficient
!     veta              Variance of viscosity -> Standard Deviation
!     vpsi1             Variance of first normal stress coefficient
!                       -> Standard Deviation
!     vpsi2             Variance of second normal stress coefficient
!                       -> Standard Deviation
```

```
! ------------------------------------------------------------------------
!     Indices
! ------------------------------------------------------------------------
!     irate             i-th shear or elongation rate
!     it                i-th time
!     itime             i-th time step
!     itra              i-th trajectory
!     ideltat           i-th deltat
! ------------------------------------------------------------------------
      use constants
      use Global
      use info
      use Numeric_kinds
      use RNGcommon
      implicit none
      integer(I4K)                           :: nu,j,k,it,it01,irate
      integer(I4K)                           :: ideltat
      integer(I4K)                           :: ndata,nt
      integer(I4K), parameter                :: msign = -1
      real(R8K)                              :: area2,volumeM2
      real(R8K)                              :: InitialArea2,InitialVal
      real(R8K)                              :: TotalArea2,TotalVal
      real(R8K)                              :: sr
      real(R8K), dimension(:),allocatable    :: rate
      real(R8K)                              :: deltat01
      real(R8K)                              :: omdth,ratedt,sq21dt
      real(R8K)                              :: q21_it01,q1122_it01
      real(R8K)                              :: q11_it01,q22_it01
      real(R8K)                              :: q11,q22,q11t01,q22t01
      real(R8K)                              :: q21,q1122,q21t01,q1122t01
      real(R8K)                              :: q3311,q3311t01 !,q2211
      real(R8K)                              :: q3311_term1,q3311_term2
      real(R8K)                              :: q11_transient,q22_transient
      real(R8K)                              :: q11_transient01,
      real(R8K)                              :: q22_transient01
      real(R8K)                              :: q21_transient,q21_transient01
      real(R8K), dimension(:),allocatable, SAVE :: aEta,vEta
      real(R8K), dimension(:),allocatable, SAVE :: aPsi1,vPsi1
      real(R8K), dimension(:),allocatable, SAVE :: mtau21,sumRyRx,sumRyRy
      real(R8K), dimension(:),allocatable, SAVE :: sumRyRxt01,sumRyRyt01
      real(R8K), dimension(:),allocatable, SAVE :: aGmod,vGmod
      real(R8K), dimension(:),allocatable, SAVE :: aGmodt01,vGmodt01
      real(R8K), dimension(:),allocatable, SAVE :: aGpsi1,vGpsi1
      real(R8K), dimension(:),allocatable, SAVE :: aGpsi1t01,vGpsi1t01
      real(R8K), dimension(:),allocatable, SAVE :: aGyra,vGyra
      real(R8K), dimension(:),allocatable, SAVE :: aGyrat01,vGyrat01
      real(R8K)                              :: qGyra,qGyrat01
      real(R8K)                              :: chi2,slope,dslope
      real(R8K), dimension(5)                :: LLS_x,LLS_y,LLS_dy
      real(R8K)                              :: ranuls,rangls
      character(len=16)                      :: itoa
      character(len=long_len)                :: file_name
      type(coord), dimension(n_bead)         :: Fphi
      !type(coord)                           :: r_c
      type(coord), dimension(n_bead)         :: R_nu, R_nu01
```

```
      type(coord), dimension(n_bead)       :: R_nuOLD, R_nuOLD01, R_t01

      call ranils()
      it_init = 1
      ndata   = n_data
      nt      = n_t

!%    [LOOP 1] Shear rate
      do irate = 1, n_rate
!        initial settings with dummy values
         call init_LL3(LLS_x,LLS_y,LLS_dy)
         slope = -1000.0D0
         dslope = -1000.0D0

!%       [LOOP A] different deltat : for extrapolation
!%       n_data loops for 'STST', one loop for 'STUP'
         if (flow_cond == 'STUP') ndata = 1
         do ideltat = 1, ndata

!%       [LOOP B] Time for Start-up flow
!%       one loop for 'STST', n_t loops for 'STUP'

         if (flow_cond == 'STST') nt = 1
         do it = it_init, nt
!%          for 'STST' time = tmax, for 'STUP' time = t(it)
            if (flow_cond == 'STUP') time = t(it)
            if (flow_cond == 'STST') time = tmax(irate)

            call deltat_test(irate,ideltat,deltat)

            sr     = rate(irate) ! for shear or shear-free

!%          n_time = floor(time/deltat)
            n_time = floor(time/deltat)
            omdth  = 0.25*deltat
!           SQRT(2kT/rho) = SQRT(2H/4H) = SQRT(0.5) = 0.707
!           where rho is friction coeff.
!           kT = H, rho = 4H
            sq21dt = SQRT(0.5D0)*(SQRT(12.0D0*deltat))
            ratedt = sr*deltat

!           [Material properties]
!           1. Viscosity: aeta
!           2. First normal stress coefficient: apsi1
!           3. Second normal stress coefficient: apsi2
!           variance of material properties : veta, vpsi1, vpsi2

!           initial setup
            if (flow_type  == 'SHEARFREE') then
               aEta(:)    = 0.0D0
               vEta(:)    = 0.0D0
            else
               aGmod(:)   = 0.0D0
               vGmod(:)   = 0.0D0
               aGpsi1(:)  = 0.0D0
```

```
               vGpsi1(:)  = 0.0D0
               !aEta(:)    = 0.0D0
               !vEta(:)    = 0.0D0
               !aPsi1(:)   = 0.0D0
               !vPsi1(:)   = 0.0D0
               !aPsi2  = 0.0D0
               !vPsi2  = 0.0D0
            end if
            mtau21(:)  = 0.0D0
            sumRyRx(:) = 0.0D0 ! SIG<RyRx>
            sumRyRy(:) = 0.0D0 ! SIG<RyRy>
            aGyra(:)   = 0.0D0
            vGyra(:)   = 0.0D0

            if (flow_cond  ==  'DISP') then
               aGmodt01(:)   = 0.0D0
               vGmodt01(:)   = 0.0D0
               aGpsi1t01(:)  = 0.0D0
               vGpsi1t01(:)  = 0.0D0
               sumRyRxt01(:) = 0.0D0 ! SIG<RyRx>
               sumRyRyt01(:) = 0.0D0 ! SIG<RyRy>
               aGyrat01(:)   = 0.0D0
               vGyrat01(:)   = 0.0D0
            end if

            TotalArea2   = 0.0D0
            TotalVal     = 0.0D0
            InitialArea2 = 0.0D0
            InitialVal   = 0.0D0

!%          [LOOP 2] for n trajectories
            do itra = 1, n_tra
!              initialize position
               call init_position()
!              connector vector Q's
               call connector(n_bead,Bead,n_tri,TriIndices,Q)
!              ------------------------------------------------
!              calculation of initial SUM(area^2) of trianlges
!              ------------------------------------------------
               InitialArea2 = area2(n_tri,Q)
!              ------------------------------------------------
!              calculation of initial volume of RBC
!              ------------------------------------------------
               if (n_bead == 4) InitialVal =  volumeM2(n_tri,Q)
!              ------------------------------------------------
!              calculation of sum of length square of triangles
!              ------------------------------------------------
               if (n_bead == 3) &
                  InitialVal = (Q(1,1) .dot. Q(1,1)) &
                             + (Q(1,2) .dot. Q(1,2)) &
                             + (Q(1,3) .dot. Q(1,3))
               ! absVEC(Q(1))**2 + absVEC(Q(2))**2 + absVEC(Q(3))**2
               ! abs(Q) is magnitude of a vector Q

!%-------------No flow condition for dimensionless time (time_equil)
```

```
!              to meet equilibrium initial condition
               call NOflow(sr,Q,Bead,InitialArea2,InitialVal)

!%-------------[LOOP 3] Time  ---------------------------------------------
               do it = 1, nt!it_init, nt
                  if (flow_cond  == 'STST') then
                     time = tmax
                  else
                     time = real(it)
                  end if


!                 Store old position at dimensionless time t and calculate
!                 R_nu=bead_nu-r_center_nu
                  if (it /= 1) then
                     do nu = 1, n_bead
                        OldBead(nu)%x = Bead(nu)%x
                        OldBead(nu)%y = Bead(nu)%y
                        OldBead(nu)%z = Bead(nu)%z
                      end do
                      call R_position(OldBead,R_nuOLD)
                  end if

               if (flow_cond  ==  'DISP' .and. it==1) it01 = 0
!%             [LOOP B] Time Integration for 1 time step with time step
!              width of deltat: Euler scheme
               do itime = 1, n_time
!                 Intermolecular Forces
!                 Each of Fphi have x, y, z component
!                 Indices 1,2,3,... represent the number of bead
                  call IntermolecularForces(Bead,Fphi)

!                 Find new position of beads
                  if (flow_type == 'SHEAR') then
!                    [DISPLACEMENT experiment]
                     if (flow_cond  == 'DISP') then
                     do nu = 1, n_bead
                        if (it == 1 .and. (itime == 1 .or. itime == 2)) &
                           then
                        ! shear strain can be induced by applying large,
                        ! constant shear rate
                        ! for a short time interval (deltat) : DPL Chap 3.4
                        ! It is applied right before t = 1. The time index
                        ! t = 1 throughout the code actually corresponds to
                        ! displacement at t > 0 in the reference.
                           NewBead(nu)%x = Bead(nu)%x +ratedt*Bead(nu)%y +&
                           ! applying large, constant shear rate; sr over
                           ! 1000 (1000*dt=0.1;10% strain)
                             omdth*Fphi(nu)%x+sq21dt*(ranuls()-0.5)
                        else
                           ! NO FLOW CONDITION
                           NewBead(nu)%x = Bead(nu)%x + &
                             omdth*Fphi(nu)%x+sq21dt*(ranuls()-0.5)
                        end if
                        NewBead(nu)%y = Bead(nu)%y + &
```

```
                              omdth*Fphi(nu)%y+sq21dt*(ranuls()-0.5)

                           NewBead(nu)%z = Bead(nu)%z + &
                                omdth*Fphi(nu)%z+sq21dt*(ranuls()-0.5)
                        end do
                        else
!                       [STEADY SHEAR FLOW experiment]
!                       note that x component has extra term.
!                       (shear flow in x-dir)
                        do nu = 1, n_bead
                           NewBead(nu)%x = Bead(nu)%x + ratedt*Bead(nu)%y + &
                           ! applying constant shear rate
                                omdth*Fphi(nu)%x+sq21dt*(ranuls()-0.5)
                           NewBead(nu)%y = Bead(nu)%y + &
                                omdth*Fphi(nu)%y+sq21dt*(ranuls()-0.5)
                           NewBead(nu)%z = Bead(nu)%z + &
                                omdth*Fphi(nu)%z+sq21dt*(ranuls()-0.5)
                        end do
                        end if
                     else if (flow_type == 'SHEARFREE') then
!                       note that z component has extra term.
                        do nu = 1, n_bead
                           NewBead(nu)%x =Bead(nu)%x-(1/2)*ratedt*Bead(nu)%x &
                                + omdth*Fphi(nu)%y+sq21dt*(ranuls()-0.5)
                           NewBead(nu)%y =Bead(nu)%y-(1/2)*ratedt*Bead(nu)%y &
                                + omdth*Fphi(nu)%z+sq21dt*(ranuls()-0.5)
                           NewBead(nu)%z = Bead(nu)%z + ratedt*Bead(nu)%z &
                                + omdth*Fphi(nu)%x+sq21dt*(ranuls()-0.5)
                        end do
                     end if

!                    translate to the origin of the reference
                     call re_position(NewBead)

!                    call constraint subroutine
                     if (constraint == 'AREA'  ) &
                        call constrnt_com_area(Q,Bead,NewBead,InitialArea2,&
                                               InitialVal,TotalArea2,TotalVal)
                     if (constraint == 'LEN'   ) &
                        call constrnt_len   (Q,Bead,NewBead,&
                                             InitialVal,TotalVal)
                     if (constraint == 'VOL'   ) &
                        call constrnt_com_vol(Q,Bead,NewBead,InitialArea2,&
                                              InitialVal,TotalArea2,TotalVal)
                     if (constraint == 'COM_no') &
                        call constrnt_com_no(Q,Bead,NewBead,InitialArea2,&
                                             InitialVal,TotalArea2,TotalVal)
                     if (constraint == 'COM_oo') &
                        call constrnt_com_oo(Q,Bead,NewBead,InitialArea2,&
                                             InitialVal,TotalArea2,TotalVal)

!                    update old position to new
                     do nu = 1, n_bead
                        Bead(nu)%x = NewBead(nu)%x
                        Bead(nu)%y = NewBead(nu)%y
```

```
                  Bead(nu)%z = NewBead(nu)%z
               end do

!              translate to the origin of the reference
               call re_position(Bead)


!              connector vector Q's
               call connector(n_bead,Bead,n_tri,TriIndices,Q)

               if (flow_cond  ==  'DISP' .and. it==1) then
               if (itime==record) then
                  it01 = it01+1
                  time01(it01) = 0.0001*itime
!                 Store position at dimensionless time 1 and
!                 calculate R_t01
                  if (it01 == 1) then
                     deltat01 = 0.0001
                     do nu = 1, n_bead
                        Bead_t01(nu)%x = Bead(nu)%x
                        Bead_t01(nu)%y = Bead(nu)%y
                        Bead_t01(nu)%z = Bead(nu)%z
                     end do
                     call R_position(Bead_t01,R_t01)
                  else
                     deltat01 = time01(it01)-time01(it01-1)
                  end if

                  call R_position(Bead,R_nu01)

               q21t01   = 0.0D0 ! yx component of stress tensor
               q11t01   = 0.0D0 ! xx component of stress tensor
               q22t01   = 0.0D0 ! yy component of stress tensor
               q1122t01 = 0.0D0 ! xx-yy component of stress tensor
               ! transient terms
               q21_transient01 = 0.0D0
               q11_transient01 = 0.0D0
               q22_transient01 = 0.0D0
               ! store values for first two time steps for
               ! 3-point differentiation
               q21_it01 = 0.0D0
               q11_it01 = 0.0D0
               q22_it01 = 0.0D0
               q1122_it01 = 0.0D0

               if (it01 == 1) then
               !<----displacement for short time (deltat) is affects
               !the material property @ itime == 1
               do nu = 1, n_bead
                  q21t01   = q21t01 + (R_nu01(nu)%y)**2
                  q11t01   = q11t01 + R_nu01(nu)%x*R_nu01(nu)%y
               end do

               q21t01   = 2.0D0 * q21t01
               q11t01   = (4.0D0/sr) * q11t01
```

```
!q2233 = 0.0D0
q1122t01 = q1122t01 + q11t01 ! - q22 (q22=0 is zero)
end if !---->

do nu = 1, n_bead
   if (it01 == 1) then
      q21_transient01 = 0.0D0
      q11_transient01 = 0.0D0
      q22_transient01 = 0.0D0
   else if (it01 == 3) then
      ! FOR FIRST TIME POINT,
      ! 3-POINT FORWARD DIFFERENCE METHOD is used.
      ! Position @ it01 == 1 is stored and
      ! time derivative term is calculated @ it01 == 3
      q21_it01 = q21_it01 + &
                 (4*R_nuOLD01(nu)%x*R_nuOLD01(nu)%y-&
                 R_nu01(nu)%x*R_nu01(nu)%y-&
                 3*R_t01(nu)%x*R_t01(nu)%y)
      q11_it01 = q11_it01 + &
                 (4*R_nuOLD01(nu)%x**2-R_nu01(nu)%x**2-&
                 3*R_t01(nu)%x**2)
      q22_it01 = q22_it01 + &
                 (4*R_nuOLD01(nu)%y**2-R_nu01(nu)%y**2-&
                 3*R_t01(nu)%y**2)
      ! 2-POINT BACKWARD DIFFERENCE METHOD is used
      q21_transient01 = q21_transient01 +&
                 (R_nu01(nu)%x*R_nu01(nu)%y- &
                 R_nuOLD01(nu)%x*R_nuOLD01(nu)%y)
      q11_transient01 = q11_transient01 + &
                 (R_nu01(nu)%x**2-R_nuOLD01(nu)%x**2)
      q22_transient01 = q22_transient01 + &
                 (R_nu01(nu)%y**2-R_nuOLD01(nu)%y**2)
   else
      ! 2-POINT BACKWARD DIFFERENCE METHOD is used
      q21_transient01 = q21_transient01 + &
                 (R_nu01(nu)%x*R_nu01(nu)%y-&
                 R_nuOLD01(nu)%x*R_nuOLD01(nu)%y)
      q11_transient01 = q11_transient01 + &
                 (R_nu01(nu)%x**2-R_nuOLD01(nu)%x**2)
      q22_transient01 = q22_transient01 + &
                 (R_nu01(nu)%y**2-R_nuOLD01(nu)%y**2)
   end if
end do

if (it01 == 3) then
   q21_it01 = msign*(1.0D0/sr)/deltat*q21_it01/deltat01
   q11_it01 = (1.0D0/sr/sr)/deltat*q11_it01/deltat01
   q22_it01 = (1.0D0/sr/sr)/deltat*q22_it01/deltat01
   q1122_it01 = q1122_it01-q11_it01+q22_it01
end if
q21_transient01 = (2.0D0/sr) &
                 / deltat*q21_transient01/deltat01
q11_transient01 = (2.0D0/sr/sr) &
                 / deltat * q11_transient01 / deltat01
q22_transient01 = (2.0D0/sr/sr) &
```

```
                                / deltat * q22_transient01 / deltat01

                  q21t01 = q21t01 - q21_transient01
                  q1122t01 = q1122t01 - q11_transient01 + q22_transient01

!                 [Material properties for Shear Flow]
                  ! Note that aGmod, vGmod, aGpsi1, vGpsi1 corresponds to
                  ! aEta, vEta, aPsi1, vPsi1 for flow_cond  /= 'DISP'
                  aGmodt01(it01)  = aGmodt01(it01) + q21t01
                  vGmodt01(it01)  = vGmodt01(it01) + q21t01*q21t01
                  aGpsi1t01(it01) = aGpsi1t01(it01) + q1122t01
                  vGpsi1t01(it01) = vGpsi1t01(it01) + q1122t01*q1122t01


                  if (it01 == 3) then
                     aGmodt01(1)  = aGmodt01(1) + q21_it01
                     vGmodt01(1)  = vGmodt01(1) + q21_it01*q21_it01
                     aGpsi1t01(1) = aGpsi1t01(1) + q1122_it01
                     vGpsi1t01(1) = vGpsi1t01(1) + q1122_it01*q1122_it01
                  end if

                     ! print sum<RxRy>,sum<RyRy> for every time step
                     do nu = 1, n_bead
                        sumRyRxt01(it01) = sumRyRxt01(it01) + &
                                           R_nu01(nu)%x*R_nu01(nu)%y
                        sumRyRyt01(it01) = sumRyRyt01(it01) + &
                                           R_nu01(nu)%y*R_nu01(nu)%y
                     end do
!                    ------------------------------------------------
!                    Radius of gyration s2 between time 0 and 1
!                    ------------------------------------------------
                     qGyrat01 = 0.0D0
                     do  nu = 1,n_bead
                        qGyrat01 = qGyrat01 + R_nu01(nu)%x**2 + &
                                   R_nu01(nu)%y**2 + R_nu01(nu)%z**2
                     end do
                     qGyrat01 = qGyrat01 / n_bead

!                    [Radius of cell]
                     aGyrat01(it01) = aGyrat01(it01) + sqrt(qGyrat01)
                     vGyrat01(it01) = vGyrat01(it01) + &
                                      sqrt(qGyrat01)*sqrt(qGyrat01)

!                 Store old position at dimensionless time t and calculate
!                 R_nu=bead_nu-r_center_nu
                     do nu = 1, n_bead
                        OldBeadt01(nu)%x = Bead(nu)%x
                        OldBeadt01(nu)%y = Bead(nu)%y
                        OldBeadt01(nu)%z = Bead(nu)%z
                     end do
                     call R_position(OldBeadt01,R_nuOLD01)
                  end if
                  end if !if (flow_cond  ==  'DISP' .and. it==1) then
```

```
                end do

              if (position) then
                   if (itra==n_tra) then !itra == 1 .or.
                      !write(ioutf5,*) 'time= ',time,'itra=',itra
!                     PRINT position of beads at each time
                     !    call display_position(ioutf3,Bead,TotalArea2,'AREA^2')
                   end if

              end if
!%            END [LOOP B]: 1 Time step Integration

!             distance from the center of mass
              call R_position(Bead,R_nu)

              if (flow_type == 'SHEAR') then
                 q21   = 0.0D0 ! yx component of stress tensor
                 q11   = 0.0D0 ! xx component of stress tensor
                 q22   = 0.0D0 ! yy component of stress tensor
                 q1122 = 0.0D0 ! xx-yy component of stress tensor
                 ! transient terms
                 q21_transient = 0.0D0
                 q11_transient = 0.0D0
                 q22_transient = 0.0D0

!%--------------- sum of each material function
!%               displacement term : -2SIG(kappa.<RR>+<RR>.kappa)
                 if (flow_cond  /= 'DISP') then
                 !<----displacement for short time (deltat) is affects
                 ! the material property @ t = 1
                 do nu = 1, n_bead
                    q21   = q21 + (R_nu(nu)%y)**2 !(Bead(nu)%y-r_c%y)**2
                    q11   = q11 + R_nu(nu)%x*R_nu(nu)%y
                 end do

                 q21   = 2.0D0 * q21
                 q11   = (4.0D0/sr) * q11
                 !q2233 = 0.0D0
                 q1122 = q1122 + q11 ! - q22 (q22=0 is zero)
                 end if !---->

!%               unsteady-state term: 2SIG(d<RR>/dt) :
!                2-POINT BACKWARD DIFFERENCE METHOD is used
                 if (flow_cond  /= 'STST') then
                 do nu = 1, n_bead
                    if (it == 1) then
                       q21_transient = q21_transient + &
                                       (R_nu(nu)%x*R_nu(nu)%y-&
                                       R_nuOLD01(nu)%x*R_nuOLD01(nu)%y)
                       q11_transient = q11_transient + (R_nu(nu)%x**2-&
                                       R_nuOLD01(nu)%x**2)
                       q22_transient = q22_transient + (R_nu(nu)%y**2- &
                                       R_nuOLD01(nu)%y**2)
                    else
                       q21_transient = q21_transient + &
```

```
                                    (R_nu(nu)%x*R_nu(nu)%y-&
                                    R_nuOLD(nu)%x*R_nuOLD(nu)%y)
                        q11_transient = q11_transient + (R_nu(nu)%x**2-&
                                    R_nuOLD(nu)%x**2)
                        q22_transient = q22_transient + (R_nu(nu)%y**2-&
                                    R_nuOLD(nu)%y**2)
                     end if
                  end do

                  if (it == 1) then
                  deltat01 = time - time01(20)
                  q21_transient = (2.0D0/sr)/deltat*q21_transient/deltat01
                  q11_transient = (2.0D0/sr/sr)/deltat*&
                                 q11_transient/deltat01
                  q22_transient = (2.0D0/sr/sr)/deltat*&
                                 q22_transient/deltat01
                  else
                  q21_transient = (2.0D0/sr) / deltat * q21_transient
                  q11_transient = (2.0D0/sr/sr) / deltat * q11_transient
                  q22_transient = (2.0D0/sr/sr) / deltat * q22_transient
                  end if

                  q21 = q21 - q21_transient
                  q1122 = q1122 - q11_transient + q22_transient

                  end if


!                 [Material properties for Shear Flow]
                  ! Note that aGmod, vGmod, aGpsi1, vGpsi1 corresponds to
                  ! aEta, vEta, aPsi1, vPsi1 for flow_cond  /= 'DISP'
                  aGmod(it)  = aGmod(it) + q21
                  vGmod(it)  = vGmod(it) + q21*q21
                  aGpsi1(it) = aGpsi1(it) + q1122
                  vGpsi1(it) = vGpsi1(it) + q1122*q1122
                  ! these value is analytically proven to be zero.
                  ! Redundant to calculate.
                  !aPsi2 = aPsi2 + q2233
                  !vPsi2 = vPsi2 + q2233*q2233
                  mtau21(it) = aGmod(it) * sr

               else if (flow_type == 'SHEARFREE') then
                     q3311 = 0.0D0 ! zz-xx component of stress tensor
                     q3311_term1 = 0.0D0 ! first term of q3311
                     q3311_term2 = 0.0D0 ! second term of q3311

!%                   sum of each material function
                     do nu = 1, n_bead
                        q3311_term1 = q3311_term1 + &
                                    (R_nu(nu)%z)**2!(Bead(nu)%z-r_c%z)**2
                        q3311_term2 = q3311_term2 + &
                                    (R_nu(nu)%x)**2!(Bead(nu)%x-r_c%x)**2
                     end do

                     q3311 = 4.0D0 * q3311_term1 + 2.0D0 * q3311_term2
```

```
                    !q2211 = 0.0D0

!                   [Material properties for Shear Flow]
                    aEta(it)  = aEta(it) + q3311
                    vEta(it)  = vEta(it) + q3311*q3311
                    !aEta2 = aEta2 + q2211
                    !vEta2 = vEta2 + q2211*q2211
                    mtau21(it) = aEta(it) * sr
                end if

                   ! print sum<RxRy>,sum<RyRy> for every time step
                   do nu = 1, n_bead
                      sumRyRx(it)   = sumRyRx(it) + &
                                      R_nu(nu)%x*R_nu(nu)%y!(Bead(nu)%x-&
                                      r_c%x)*(Bead(nu)%y-r_c%y)
                      sumRyRy(it)   = sumRyRy(it) + &
                                      R_nu(nu)%y*R_nu(nu)%y!(Bead(nu)%x-&
                                      r_c%y)*(Bead(nu)%y-r_c%y)
                   end do

!                  ------------------------------------------------
!                  Radius of gyration s2
!                  ------------------------------------------------
                   qGyra = 0.0D0
                   do  nu = 1,n_bead
                      qGyra = qGyra + R_nu(nu)%x**2 + R_nu(nu)%y**2 + &
                              R_nu(nu)%z**2!R_nu(nu) .dot. R_nu(nu)
                   end do
                   qGyra = qGyra / n_bead

!                  [Radius of cell]
                   aGyra(it) = aGyra(it) + sqrt(qGyra)
                   vGyra(it) = vGyra(it) + sqrt(qGyra)*sqrt(qGyra)


                end do
!%              END [LOOP 3]: Time

             end do
!%           END [LOOP 2]: itra

!%           Average over number of trajectories
!            time between 0 and 1 : for displacement experiment only
             if (flow_cond  ==  'DISP') then
                do j=1,20
                   aGmodt01(j)  = aGmodt01(j) / n_tra
                   vGmodt01(j)  = vGmodt01(j) / n_tra
                   vGmodt01(j)  = sqrt((vGmodt01(j)-aGmodt01(j)*&
                                  aGmodt01(j)) / (n_tra-1))
                   aGpsi1t01(j) = aGpsi1t01(j) / n_tra
                   vGpsi1t01(j) = vGpsi1t01(j) / n_tra
                   vGpsi1t01(j) = sqrt((vGpsi1t01(j)-&
                                  aGpsi1t01(j)*aGpsi1t01(j)) / (n_tra-1))

                   sumRyRxt01(j) = sumRyRxt01(j) / n_tra
```

```
                  sumRyRyt01(j) = sumRyRyt01(j) / n_tra
                  aGyrat01(j) = aGyrat01(j) / n_tra
                  vGyrat01(j) = vGyrat01(j) / n_tra
                  vGyrat01(j) = sqrt((vGyrat01(j)-&
                                aGyrat01(j)*aGyrat01(j))/(n_tra-1))
               end do
            end if
!           time between 1 and nt
            do k=1,nt
               if (flow_type  == 'SHEARFREE') then
                  aEta(k)  = aEta(k) / n_tra
                  vEta(k)  = vEta(k) / n_tra
                  vEta(k)  = sqrt((vEta(k)-aEta(k)*aEta(k)) / (n_tra-1))
               else
               !if (flow_cond  ==  'DISP') then
                  aGmod(k)  = aGmod(k) / n_tra
                  vGmod(k)  = vGmod(k) / n_tra
                  vGmod(k)  = sqrt((vGmod(k)-aGmod(k)*aGmod(k)) / (n_tra-1))
                  aGpsi1(k) = aGpsi1(k) / n_tra
                  vGpsi1(k) = vGpsi1(k) / n_tra
                  vGpsi1(k) = sqrt((vGpsi1(k)-aGpsi1(k)*aGpsi1(k)) / &
                              (n_tra-1))
               !else
                  !aEta(k)  = aEta(k) / n_tra
                  !vEta(k)  = vEta(k) / n_tra
                  !vEta(k)  = sqrt((vEta(k)-aEta(k)*aEta(k)) / (n_tra-1))
                  !aPsi1(k) = aPsi1(k) / n_tra
                  !vPsi1(k) = vPsi1(k) / n_tra
                  !vPsi1(k) = sqrt((vPsi1(k)-aPsi1(k)*aPsi1(k)) / (n_tra-1))
               !end if
               end if
               mtau21(k) = mtau21(k) / n_tra
               sumRyRx(k) = sumRyRx(k) / n_tra
               sumRyRy(k) = sumRyRy(k) / n_tra
               aGyra(k) = aGyra(k) / n_tra
               vGyra(k) = vGyra(k) / n_tra
               vGyra(k) = sqrt((vGyra(k)-aGyra(k)*aGyra(k))/ (n_tra-1))
            end do


!%          Display results
!           print result of time between 1 and nt
            do k=1,nt
              !if (flow_type == 'SHEAR') then
                  call display_results(ioutf1,sr,deltat,&
                            aGmod(k),vGmod(k),aGpsi1(k),&
                            vGpsi1(k),n_time,time,sumRyRx(k),sumRyRy(k),&
                            aGyra(k),vGyra(k))
            end do

!%       END [LOOP A]: ideltat

!%       Extrapolated material properties for steady state
100      continue
      end do
```

```
!%    END [LOOP 1]: irate
!%       simmulation time recorded
      stop
   end subroutine FLOW
!  ------------------------------------------------------------------------
!@ subroutine constrnt_com_no - calculate LaGrange multiplier
!      Program (steps procedure)that calculates the lambda coefficient that
!      adds the constraint force and calculates the corrected positions to
!      satisfy the homologic constraint.
   subroutine constrnt_com_no
              (Qi,point,Newpoint,TotArea2,TotLen,TriArea2,TriLen)
      use constants
      use Global
      use Numeric_kinds
      use vec_func
      implicit none
      real(R8K), intent(IN)                      :: TotArea2,TotLen
      real(R8K), intent(OUT)                     :: TriArea2,TriLen
      type(coord), dimension(n_bead), intent(INOUT)   :: point
      type(coord), dimension(n_bead), intent(OUT)     :: Newpoint
      type(coord), dimension(*), intent(INOUT)        :: Qi
!     Local variables:
      integer(I4K)                               :: iter,nu
      real(R8K), dimension(2,2)                  :: g,modG
      real(R8K)                                  :: detG
!     real(R8K)                                  :: discr1,discr2
      real(R8K)                                  :: sigma1,sigma2
      real(R8K)                                  :: lambda1,lambda2
      real(R8K), dimension(n_bead,n_bead+1)      :: A
      real(R8K)                                  :: area2
      real(R8K)                                  :: param,param1,param2
      type(coord), dimension(n_bead)             :: ds1point,ds2point
      type(coord), dimension(n_bead)             :: ds1Newpoint,ds2Newpoint
!     center of mass
      call center_mass(point)

!     translate center of mass to the origin of the reference
      do nu = 1, n_bead
         point(nu)%x = point(nu)%x-r_c%x
         point(nu)%y = point(nu)%y-r_c%y
         point(nu)%z = point(nu)%z-r_c%z
      end do
!     ---------- sigma1 : area constraint ---------------------------------
!     DEL(sigma) @t : not scalar
!     compute the partial derivatives of the area constraint
!     using the old position
      param = 0.5D0
      call scalarcomp(Qi,A)
      do nu = 1, n_bead
         ds1point(nu) = param .M. ((A(nu,1).M.Qi(1)) .A. (A(nu,2).M.Qi(2)))
      end do
!     ---------- sigma2 : sum of length square constraint ---------------
!     DEL(sigma) @t : not scalar
!     compute the partial derivatives of the sum of length square
!     constraint using the old position
```

```
      param1 = 2.
      param2 = -2.
      ds2point(1) = param2 .M. (Qi(1) .A. Qi(2))
      ds2point(2) = param1 .M. (Qi(1) .S. Qi(3))
      ds2point(3) = param1 .M. (Qi(2) .A. Qi(3))

      lambda1 = 0.0D0
      lambda2 = 0.0D0

      do iter = 1, maxiter
!        discr1 = 0.0D0 discr2 = 0.0D0
       call connector(Newpoint,Qi)
!        ============================================================
!        compute the constraint SIG using new position
!        ---------- sigma1 : area constraint -----------------------------
         TriArea2 = area2(Qi(1),Qi(2))
!        TotArea2 is four equilateral triangle with each area = 10
         sigma1 = TriArea2 - TotArea2
!        ---------- sigma2 : sum of length square constraint -------------
         TriLen = (Qi(1).dot.Qi(1))+(Qi(2).dot.Qi(2))+(Qi(3).dot.Qi(3))
         sigma2 = TriLen - TotLen
!        ============================================================
         if (abs(sigma1) .LT. tol .and. abs(sigma2) .LT. tol) goto 200
!        if (discr1 .LT. abs(sigma1)) discr1 = abs(sigma1)
!        if (discr2 .LT. abs(sigma2)) discr2 = abs(sigma2)
!        ============================================================
!        DEL(sigma) @(t+deltat) :
!        compute the partial derivatives of the constraint using
!        the New position
!        scalar values for new position with updated Qi
!        ---------- sigma1 : area constraint -----------------------------
         call scalarcomp(Qi,A)
         do nu = 1, n_bead
            ds1Newpoint(nu) =
            param.M.((A(nu,1).M.Qi(1)).A.(A(nu,2).M.Qi(2)))
         end do
!        ---------- sigma2 : sum of length square constraint -------------
         ds2Newpoint(1) = param2 .M. (Qi(1) .A. Qi(2))
         ds2Newpoint(2) = param1 .M. (Qi(1) .S. Qi(3))
         ds2Newpoint(3) = param1 .M. (Qi(2) .A. Qi(3))
!        ============================================================
!        calculate modified metric matrix
         modG = 0.
         do nu = 1, n_bead
            modG(1,1) = modG(1,1) + (ds1Newpoint(nu) .dot. ds1point(nu))
            modG(1,2) = modG(1,2) + (ds1Newpoint(nu) .dot. ds2point(nu))
            modG(2,1) = modG(2,1) + (ds2Newpoint(nu) .dot. ds1point(nu))
            modG(2,2) = modG(2,2) + (ds2Newpoint(nu) .dot. ds2point(nu))
         end do

!        This is for [2x2] matrix
!        put algorithm to inverse matrix modG to get [ d' x d' ]
!        metric matrix g
         detG = modG(1,1)*modG(2,2)-modG(1,2)*modG(2,1)
         g(1,1) = modG(2,2)/detG
```

```
         g(1,2) = -modG(1,2)/detG
         g(2,1) = -modG(2,1)/detG
         g(2,2) = modG(1,1)/detG

!        Lagrange multiplier
!        lambda = (rho)* g(j,k)*sigma @(t+deltat)
         lambda1 = lambda1 + g(1,1)*sigma1 + g(1,2)*sigma2
         lambda2 = lambda2 + g(2,1)*sigma1 + g(2,2)*sigma2

!        r_CON = r_UN - (1/rho)*lambda * DEL(sigma) @t : see Ottinger
         do nu = 1, n_bead
            Newpoint(nu) = Newpoint(nu) .S. &
                    ((lambda1 .M. ds1point(nu)).A.(lambda2.M.ds2point(nu)))
         end do

!        center of mass
         call center_mass(Newpoint)

!        translate center of mass to the origin of the reference
         do nu = 1, n_bead
            Newpoint(nu)%x = Newpoint(nu)%x-r_c%x
            Newpoint(nu)%y = Newpoint(nu)%y-r_c%y
            Newpoint(nu)%z = Newpoint(nu)%z-r_c%z
         end do

!        if discr=0, end of iteration
200      if (abs(sigma1) .LT. tol .and. abs(sigma2) .LT. tol) return

      end do
      return
   end subroutine constrnt_com_no
!  ------------------------------------------------------------------------
!@ subroutine constrnt_com_oo - calculate LaGrange multiplier
!      Program (steps procedure)that calculates the lambda coefficient that
!      adds the constraint force and calculates the corrected positions
!      to satisfy the homologic constraint.
!      USE ONLY OLD POSITION TO CALCULATE METRIC MATRIX
!      c = c' = 0
   subroutine constrnt_com_oo
              (Qi,point,Newpoint,TotArea2,TotLen,TriArea2,TriLen)
      use constants
      use Global
      use Numeric_kinds
      use vec_func
      implicit none
      real(R8K), intent(IN)                    :: TotArea2,TotLen
      real(R8K), intent(OUT)                   :: TriArea2,TriLen
      type(coord), dimension(n_bead), intent(INOUT)      :: point
      type(coord), dimension(n_bead), intent(OUT)        :: Newpoint
      type(coord), dimension(*), intent(INOUT)           :: Qi
!     Local variables:
      integer(I4K)                             :: iter,nu
      real(R8K), dimension(2,2)                :: g,modG
      real(R8K)                                :: detG
      real(R8K)                                :: discr1,discr2
```

```
      real(R8K)                                   :: sigma1,sigma2
      real(R8K)                                   :: lambda1,lambda2
      real(R8K), dimension(n_bead,n_bead+1)       :: A
      real(R8K)                                   :: area2
      real(R8K)                                   :: param,param1,param2
      type(coord), dimension(n_bead)              :: ds1point,ds2point
      type(coord), dimension(n_bead)              :: ds1Newpoint,ds2Newpoint

!     center of mass
      call center_mass(point)

!     translate center of mass to the origin of the reference
      do nu = 1, n_bead
         point(nu)%x = point(nu)%x-r_c%x
         point(nu)%y = point(nu)%y-r_c%y
         point(nu)%z = point(nu)%z-r_c%z
      end do
!     ---------- sigma1 : area constraint ---------------------------------
!     DEL(sigma) @t : not scalar
!     compute the partial derivatives of the area constraint using the
!     old position
      param = 0.5D0
      call scalarcomp(Qi,A)
      do nu = 1, n_bead
         ds1point(nu) = param .M. ((A(nu,1).M.Qi(1)) .A. (A(nu,2).M.Qi(2)))
      end do
!     ---------- sigma2 : sum of length square constraint ---------------
!     DEL(sigma) @t : not scalar
!     compute the partial derivatives of the sum of length square
!     constraint using the old position
      param1 = 2.0D0
      param2 = -2.0D0
      ds2point(1) = param2 .M. (Qi(1) .A. Qi(2))
      ds2point(2) = param1 .M. (Qi(1) .S. Qi(3))
      ds2point(3) = param1 .M. (Qi(2) .A. Qi(3))

!     calculate modified metric matrix
      modG = 0.
      do nu = 1, n_bead
         modG(1,1) = modG(1,1) + (ds1point(nu) .dot. ds1point(nu))
         modG(1,2) = modG(1,2) + (ds1point(nu) .dot. ds2point(nu))
         modG(2,1) = modG(2,1) + (ds2point(nu) .dot. ds1point(nu))
         modG(2,2) = modG(2,2) + (ds2point(nu) .dot. ds2point(nu))
      end do

!     This is for [2x2] matrix
!     put algorithm to inverse matrix modG to get [ d' x d' ]
!     metric matrix g
      detG = modG(1,1)*modG(2,2)-modG(1,2)*modG(2,1)
      g(1,1) = modG(2,2)/detG
      g(1,2) = -modG(1,2)/detG
      g(2,1) = -modG(2,1)/detG
      g(2,2) = modG(1,1)/detG

      lambda1 = 0.0D0
```

```
      lambda2 = 0.0D0

      do iter = 1, maxiter
         discr1 = 0.0D0
         discr2 = 0.0D0
         call connector(Newpoint,Qi)
!        ============================================================
!        compute the constraint SIG using new position
!        ---------- sigma1 : area constraint -----------------------------
         TriArea2 = area2(Qi(1),Qi(2))
!        TotArea2 is four equilateral triangle with each area = 10
         sigma1 = TriArea2 - TotArea2
!        ---------- sigma2 : sum of length square constraint -------------
         TriLen = (Qi(1).dot.Qi(1)+(Qi(2).dot.Qi(2))+(Qi(3).dot.Qi(3))
         sigma2 = TriLen - TotLen
!        ============================================================
         if (abs(sigma1) .LT. tol .and. abs(sigma2) .LT. tol) goto 200
         if (discr1 .LT. abs(sigma1)) discr1 = abs(sigma1)
         if (discr2 .LT. abs(sigma2)) discr2 = abs(sigma2)

!        Lagrange multiplier
!        lambda = (rho)* g(j,k)*sigma @(t+deltat)
         lambda1 = lambda1 + g(1,1)*sigma1 + g(1,2)*sigma2
         lambda2 = lambda2 + g(2,1)*sigma1 + g(2,2)*sigma2

!        r_CON = r_UN - (1/rho)*lambda * DEL(sigma) @t : see Ottinger
         do nu = 1, n_bead
            Newpoint(nu) = Newpoint(nu) .S. &
                        ((lambda1.M.ds1point(nu)).A.(lambda2.M.ds2point(nu)))
         end do

!        center of mass
         call center_mass(Newpoint)

!        translate center of mass to the origin of the reference
         do nu = 1, n_bead
            Newpoint(nu)%x = Newpoint(nu)%x-r_c%x
            Newpoint(nu)%y = Newpoint(nu)%y-r_c%y
            Newpoint(nu)%z = Newpoint(nu)%z-r_c%z
         end do

!        if discr=0, end of iteration
200       if (discr1 .LT. tol .and. discr2 .LT. tol) return
      end do
      return
   end subroutine constrnt_com_oo
!  ------------------------------------------------------------------------
!@ subroutine constrnt_area_no - calculate Lagrange multiplier
   subroutine  constrnt_area(triIndices,Qm,point,Newpoint,InitArea2,TotArea2)
      use Global
      use Numeric_kinds
      use vec_func
      implicit none
      integer(I4K), dimension(n_tri,3), intent(IN)        :: triIndices
      real(R8K), intent(IN)                               :: InitArea2
```

```
      real(R8K), intent(OUT)                                :: TotArea2
      type(coord), dimension(n_bead), intent(INOUT)         :: point
      type(coord), dimension(n_bead), intent(INOUT)         :: Newpoint
      type(coord), dimension(n_tri,3), intent(INOUT)        :: Qm
      ! Local variables:
      integer(I4K)                                  :: iter,nu
      real(R8K)                                     :: param
      real(R8K)                                     :: sigma1,discr
      real(R8K)                                     :: denom,lambda
      real(R8K), dimension(2,2)                     :: g,modG
      real(R8K)                                     :: area2
      type(coord), dimension(n_bead)                :: ds1point
      type(coord), dimension(n_bead)                :: ds1Newpoint

      param = 0.5D0
!     metric matrix. NOT used in this subroutine
      modG = 0.
      g = 0.

!     DEL(sigma) @t : not scalar
!     Initialization
      do nu = 1, n_bead
         ds1point(nu)%x = 0.0D0
         ds1point(nu)%y = 0.0D0
         ds1point(nu)%z = 0.0D0
      end do
      do nu = 1, n_bead
         ds1Newpoint(nu)%x = 0.0D0
         ds1Newpoint(nu)%y = 0.0D0
         ds1Newpoint(nu)%z = 0.0D0
      end do

!     compute the partial derivatives of the constraint (ds1point)
!     using the old position
      call crossProductQ1Q2(n_bead,n_tri,triIndices,Qm,ds1point)

      do iter = 1, maxiter
         discr = 0.0D0
!        connector vector Q's
         call connector(n_bead,Newpoint,n_tri,triIndices,Qm)

         TotArea2 = area2(n_tri,Qm)

!        compute the constraint SIG using new position
!        InitArea2 is four equilateral Totangle with each area
         sigma1 = TotArea2 - InitArea2

         if (abs(sigma1) .LT. tol) goto 200 !sqrt(abs(sigma1))
         if (discr .LT. abs(sigma1)) discr = abs(sigma1)

!        DEL(sigma) @(t+deltat) :
!        compute the partial derivatives of the constraint using
!        the New position
!        scalar values for new position with updated Qm
         call crossProductQ1Q2(n_bead,n_tri,triIndices,Qm,ds1Newpoint)
```

```
!        calculate denominator
         denom = 0.0D0
         do nu = 1, n_bead
            denom = denom + (ds1Newpoint(nu).dot.ds1point(nu))
         end do

!        Lagrange multiplier
!        lambda = (rho/deltat)* sigma1 @(t+deltat)/denom
!        denom = SUM(DEL(sigma1) @(t+deltat) .dot. DEL(sigmaCON) @t)
         lambda = sigma1/denom

!        r_new = r_old - (deltat/rho)*lambda * DEL(sigmaCON) @t :
         do nu = 1, n_bead
            Newpoint(nu) = Newpoint(nu) .S. (lambda .M. ds1point(nu))
         end do

!        if discr=0, end of iteration
200      if (discr .LT. tol) then
            return
         end if
      end do
      return
   end subroutine constrnt_area
!  ------------------------------------------------------------------------
!@ subroutine center_mass - Calculate center of mass
   subroutine center_mass(point,r_center)
      use Global
      use Numeric_kinds
      use vec_func
      implicit none
! ------------------------------------------------------------------------
!     The variables used in this code are:
! ------------------------------------------------------------------------
!     point                  Position of bead, vector
!     r_center               Center of mass of position of bead, vector
! ------------------------------------------------------------------------
      type(coord), intent(IN), dimension(n_bead) :: point
      type(coord), intent(OUT)                   :: r_center
      ! Local variables:
      integer(I4K)                               :: nu

      r_center%x = 0
      r_center%y = 0
      r_center%z = 0

      do nu = 1, n_bead
         r_center%x = r_center%x + point(nu)%x
         r_center%y = r_center%y + point(nu)%y
         r_center%z = r_center%z + point(nu)%z
      end do

      r_center%x = r_center%x / n_bead
      r_center%y = r_center%y / n_bead
      r_center%z = r_center%z / n_bead
```

```
   end subroutine center_mass
!  ------------------------------------------------------------------------
!@ subroutine re_position - reposition of the beads to the origin of the
!                          reference
   subroutine re_position(point)
      use Global
      use Numeric_kinds
      use vec_func
      implicit none
      type(coord), intent(INOUT), dimension(n_bead)      :: point
      ! Local variables:
      integer(I4K)                                       :: nu
      type(coord)                                        :: r_center

      call center_mass(point,r_center)
      do nu = 1, n_bead
!        point(nu) = point(nu) .S. r_center
         point(nu)%x = point(nu)%x-r_center%x
         point(nu)%y = point(nu)%y-r_center%y
         point(nu)%z = point(nu)%z-r_center%z
      end do

   end subroutine re_position
! ------------------------------------------------------------------------
!@ subroutine R_position - Calculate distance from the center of mass
   subroutine R_position(point,R_nu)
      use Global
      use Numeric_kinds
      use vec_func
      implicit none
      type(coord), intent(IN), dimension(n_bead)         :: point
      type(coord), intent(OUT), dimension(n_bead)        :: R_nu
      ! Local variables:
      integer(I4K)                                       :: nu
      type(coord)                                        :: r_center

      call center_mass(point,r_center)
      do nu = 1, n_bead
!        R_nu(nu) = point(nu) .S. r_center
         R_nu(nu)%x = point(nu)%x-r_center%x
         R_nu(nu)%y = point(nu)%y-r_center%y
         R_nu(nu)%z = point(nu)%z-r_center%z
      end do

   end subroutine R_position
!  ------------------------------------------------------------------------
!@ subroutine connector - connector vector for each local Triangle
   subroutine connector(n_bead,point,n_tri,triIndices,Qm)
      use Numeric_kinds
      use vec_func
      implicit none
! ------------------------------------------------------------------------
!     The variables used in this code are:
! ------------------------------------------------------------------------
```

```
!     triIndices                    Indices of each triangle
! ------------------------------------------------------------------------
      integer(I4K), intent(IN)                              :: n_tri,n_bead
      integer(I4K), dimension(n_tri,3), intent(IN)          :: triIndices
      type(coord), dimension(n_bead), intent(IN)            :: point
      type(coord), dimension(n_tri,3), intent(OUT)          :: Qm
      integer(I4K)                                          :: m

!     Q(n_tri,1) = r(n_tri,2) - r(n_tri,1)
!     Q(n_tri,2) = r(n_tri,3) - r(n_tri,1)
!     Q(n_tri,3) = r(n_tri,3) - r(n_tri,2)
!     Only need set of two connector vectors ( Qm(m,1) and Qm(m,2) )
!     for each triangle to calculate the area
!     However, need all three sets to get Inter-particle Forces.
!     Details are in subroutine UNIQUEconnector.
      do m = 1,n_tri
         Qm(m,1) = point(triIndices(m,2)) .S. point(triIndices(m,1))
         Qm(m,2) = point(triIndices(m,3)) .S. point(triIndices(m,1))
         Qm(m,3) = point(triIndices(m,3)) .S. point(triIndices(m,2))
      end do

   end subroutine connector
!  ------------------------------------------------------------------------
!@  subroutine  UNIQUEconnector  -  Unique  connector  vectors  from  sets  of
connector vectors
   subroutine UNIQUEconnector(n_tri,triIndices,count,uniquepairIndices)
      use Numeric_kinds
      use vec_func
      implicit none
! ------------------------------------------------------------------------
!     The variables used in this code are:
! ------------------------------------------------------------------------
!     triIndices             Indices of each triangle
!     uniquepairIndices      Unique pair out of pairnodes
!     pairnodes1             1st pairnodes for each row (triangle)
!     pairnodes2             2nd pairnodes for each row (triangle)
!     pairnodes3             3rd pairnodes for each row (triangle)
!     pairnodes              All combination of two indicies
!     count                  Count of unique pair indices
!     isUnique               Boolean to check uniqueness
! ------------------------------------------------------------------------
      integer(I4K), intent(IN)                          :: n_tri
      integer(I4K), intent(OUT)                         :: count
      integer(I4K), dimension(n_tri,3), intent(IN)      :: triIndices
      integer(I4K), dimension(n_tri*3,2), intent(OUT)   :: uniquepairIndices
      integer(I4K)                                      :: k,l,m
      integer, dimension(n_tri,2)                       :: pairnodes1
      integer, dimension(n_tri,2)                       :: pairnodes2
      integer, dimension(n_tri,2)                       :: pairnodes3
      integer(I4K), dimension(n_tri*3,2)                :: pairnodes
      LOGICAL,DIMENSION(n_tri*3)                        :: isUnique

      do m = 1, n_tri
         pairnodes1(m,1) = triIndices(m,1)
         pairnodes1(m,2) = triIndices(m,2)
```

```
         pairnodes2(m,1) = triIndices(m,1)
         pairnodes2(m,2) = triIndices(m,3)
         pairnodes3(m,1) = triIndices(m,2)
         pairnodes3(m,2) = triIndices(m,3)
      end do

      pairnodes(:,1) = [pairnodes1(:,1), pairnodes2(:,1), pairnodes3(:,1)]
      pairnodes(:,2) = [pairnodes1(:,2), pairnodes2(:,2), pairnodes3(:,2)]

      call SORTpairnodes(n_tri,pairnodes)

      isUnique = .False.
      count = 0
      uniquepairIndices(:,:) = 0

      do l = 1,n_tri*3
         if (l == 1) then
            isUnique(l) = .TRUE.
            count = count+1
            uniquepairIndices(count,1) = pairnodes(l,1)
            uniquepairIndices(count,2) = pairnodes(l,2)
         else
            do k = 1, count
               if ((pairnodes(l,1)==uniquepairIndices(k,1)) .and. &
                   (pairnodes(l,2)==uniquepairIndices(k,2))) then
                     isUnique(l) = .FALSE.
                     exit
               else
                     isUnique(l) = .TRUE.
               end if
            end do
            if (isUnique(l)) then
               isUnique(l) = .TRUE.
               count = count+1
               uniquepairIndices(count,1) = pairnodes(l,1)
               uniquepairIndices(count,2) = pairnodes(l,2)
            end if
         end if
      end do

   end subroutine UNIQUEconnector
!  ------------------------------------------------------------------------------
!@ subroutine SORTpairnodes - sort the row of each pair nodes
   subroutine SORTpairnodes(n_tri,pairnodes)
      use Numeric_kinds
      use vec_func
      implicit none
! ------------------------------------------------------------------------
!     The variables used in this code are:
! ------------------------------------------------------------------------
!     pairnodes              All combination of two indicies
!     tempint                Temporary storage for integer
! ------------------------------------------------------------------------
      integer(I4K), intent(IN)                            :: n_tri
      integer(I4K), dimension(n_tri*3,2), intent(INOUT)   :: pairnodes
```

```
      integer(I4K)                                          :: m
      integer(I4K)                                          :: tempint

      do m = 1, n_tri*3
         if (pairnodes(m,1) .GT. pairnodes(m,2)) then
            tempint = pairnodes(m,1)
            pairnodes(m,1) = pairnodes(m,2)
            pairnodes(m,2) = tempint
         end if
      end do
   end subroutine SORTpairnodes
!  ------------------------------------------------------------------------
!@ subroutine InterParticleForces - Inter-particle Forces for each points
   subroutine InterParticleForces(point,triIndices,Fphi)
      use Global
      use Numeric_kinds
      use vec_func
      implicit none
! ------------------------------------------------------------------------
!     The variables used in this code are:
! ------------------------------------------------------------------------
!     triIndices             Indices of each triangle
!     uniquepairIndices      Unique pair out of pairnodes
!     count                  Count of unique pair indices
!     Qunique                Connector vectors of unique pair indices
!                            type(coord) and has dimension of (count)
!                            This is equivalent to Qk in the dissertation.
!     Fphi                   Inter-particle Forces
! ------------------------------------------------------------------------
      integer(I4K), dimension(n_tri,3), intent(IN)     :: triIndices
      type(coord), dimension(n_bead), intent(IN)       :: point
      type(coord), dimension(n_bead), intent(OUT)      :: Fphi
      integer(I4K)                                     :: nu,k,count
      integer(I4K), dimension(n_tri*3,2)               :: uniquepairIndices

      call UNIQUEconnector(n_tri,triIndices,count,uniquepairIndices)

      if (.not. allocated(Qunique)) allocate(Qunique(count))
      do k = 1, count
         Qunique(k) = point(uniquepairIndices(k,2)) .S.
                      point(uniquepairIndices(k,1))
      end do

      do nu = 1, n_bead
         Fphi(nu)%x = 0.0D0
         Fphi(nu)%y = 0.0D0
         Fphi(nu)%z = 0.0D0
      end do

      do nu = 1, n_bead
         do k = 1, count
            if (uniquepairIndices(k,1) == nu) then
               Fphi(nu) = Fphi(nu) .A. Qunique(k)
            end if
            if (uniquepairIndices(k,2) == nu) then
```

```
               Fphi(nu) = Fphi(nu) .S. Qunique(k)
            end if
         end do
      end do

   end subroutine InterParticleForces
! ------------------------------------------------------------------------
!@ subroutine crossProductQ1Q2 - scalar component of ds1point
!@                               (area constraint) for triangles
   subroutine crossProductQ1Q2(n_bead,n_tri,triIndices,Qm,delSIG)
      use Numeric_kinds
      use vec_func
      implicit none
      integer(I4K), INTENT(IN)                              :: n_bead,n_tri
      integer(I4K), dimension(n_tri,3), intent(IN)      :: triIndices
      type(coord), dimension(n_tri,3), intent(IN)       :: Qm
      type(coord), dimension(n_bead), intent(OUT)       :: delSIG
      ! Local variables:
      integer(I4K)                                          :: nu,m
      real(R8K)                                             :: delA,delB,delC
      real(R8K), parameter                                  :: minus  = -1.0D0
      real(R8K), parameter                                  :: plus   = 1.0D0
      real(R8K), parameter                                  :: zero   = 0.0D0
      real(R8K), dimension(n_bead)                          :: delQ1,delQ2
      real(R8K)                                             :: param
      real(R8K)                                             :: Q1Q1,Q1Q2,Q2Q2

      param = 0.5D0

      do nu = 1, n_bead
         delSIG(nu)%x = 0.0D0
         delSIG(nu)%y = 0.0D0
         delSIG(nu)%z = 0.0D0
      end do

      ! A,B,C denotes the three indices for each triangle that is stored
      ! in triIndices.
      ! Originally read from Excel file with A,B,C columns.

      do m = 1,n_tri ! for each triangle
         ! w = [Q1 X Q2]
         !w(m) = Qm(m,1) .X. Qm(m,2)

         ! A(m) = triIndices(m,1)
         ! B(m) = triIndices(m,2)
         ! C(m) = triIndices(m,3)
         delQ1(:) = 0
         delQ2(:) = 0

         Q1Q1 = Qm(m,1) .dot. Qm(m,1)
         Q1Q2 = Qm(m,1) .dot. Qm(m,2)
         ! Q1Q2 = Q2Q1
         Q2Q2 = Qm(m,2) .dot. Qm(m,2)

         do nu = 1,n_bead
```

```
            delA = 0
            delB = 0
            delC = 0
            if ( triIndices(m,1) == nu ) then
               delA = minus
            else
               delA = zero
            end if
            if ( triIndices(m,2) == nu ) then
               delB = plus
            else
               delB = zero
            end if
            if ( triIndices(m,3) == nu ) then
               delC = plus
            else
               delC = zero
            end if

            ! delw = {delQ1 X Q2 - delQ2 X Q1}
           !delw(m,nu) = (delB+delA) .dot. Q(m,2) - (delC+delA) .dot. Q(m,1)
            delQ1(nu) = delB + delA ! A(m) /= B(m)
            delQ2(nu) = delC + delA ! A(m) /= C(m)
            ! Indices of each triangle is never the same number
            ! Therefore, delQ1(nu)s and delQ2(nu)s are always -1,1,or 0

            ! DEL(sigma) = 0.5 * SUM([delw .dot. w])
            delSIG(nu) = delSIG(nu) .A. &
                         ((delQ1(nu) .M. ((Q2Q2 .M. Qm(m,1)) .S. (Q1Q2 .M.
                           Qm(m,2)))) .S. (delQ2(nu) .M. &
                         ((Q1Q2 .M. Qm(m,1)) .S. (Q1Q1 .M. Qm(m,2)))))
         end do
      end do

      do nu = 1, n_bead
         delSIG(nu) = param .M. delSIG(nu)
      end do

   end subroutine crossProductQ1Q2
!  ------------------------------------------------------------------------
!@ function area2 - Calculate (area)^2 of a triangle
   function area2(n_tri,Qm)
      use Numeric_kinds
      use vec_func
      implicit none
! ------------------------------------------------------------------------
!     The variables used in this code are:
! ------------------------------------------------------------------------
!     Qm1,Qm2               Vector
!     area2                 area**2 of triangle
! ------------------------------------------------------------------------
      integer(I4K), intent(in)                                  :: n_tri
      type(coord), dimension(n_tri,3), intent(IN)               :: Qm
      ! Local variables:
      integer(I4K)                                              :: m
```

```
      real(R8K)                                                    :: area2

!     area  = 1/2 * SQRT(SUM(Q1Q2 .X. Q1Q2))
!           = 1/2 * abs((Q1 .X. Q2))
!     area2 = 1/4 * (Q1 .X. Q2)**2
      area2 = 0.0D0
      do m = 1,n_tri
         area2 = area2+((Qm(m,1) .X. Qm(m,2)) .dot. (Qm(m,1) .X. Qm(m,2)))
      end do
      area2 = 1/4. * area2

   end function area2
!  ------------------------------------------------------------------------
```

VITA

Kyung-Hyo Kim

1255 S.State St.
Chicago, IL 60605

Phone: +1 (773) 398-7721
Email: kim.kyunghyokim@gmail.com

EDUCATION

Ph.D., Chemical Engineering, University of Illinois at Chicago, Chicago, Illinois, 2015
B.A., Chemistry, Soongsil University, Seoul, Korea, 2001

RESEARCH EXPERIENCE

Department of Chemical Engineering, 2008-Present
University of Illinois at Chicago, Chicago, IL
Dissertation: "A Stochastic Sumulation Method Using Constraints for the Modeling of Blood Rheology"
Institute for Health Research and Policy, 2009-Present
University of Illinois at Chicago, Chicago, IL

- Research project on the development of advanced statistical software tools for health and prevention studies: Maximum marginal likelihood estimates for mixed-effects ordinal probit, logistic, and complementary log-log regression models with operating characteristic (ROC).

Department of Chemical Engineering, 06/2008 - 12/2008
University of Illinois at Chicago, Chicago, IL

- Research on the characterization of the metal surface to develop high quality coating with the enhancement of material properties.

Department of Chemistry, 08/2002 – 12/2002
University of Illinois at Chicago, Chicago, IL

- Research on analytical instrumentation and methods for neurological system: A push-pull perfusion method.

Department of Chemistry, 03/2000 – 02/2001
SoongSil University, Seoul, Korea

- Undergraduate research: Synthesis of Supramolecular based on Cyclotriveratrylene (CTV)
- Dissertation: "Sensors and Switches in Supramolecular Chemistry"

TEACHING EXPERIENCE

Undergraduate Research Advisor, 09/2008 - 08/2011
University of Illinois at Chicago

- Xin Qin, "Continued Simulation of three Bead-Spring Rings with a constant area constaint"
- Vijeta Patel, "Applying Volume Constraint to Red Blood Cell Model"

Teaching Assistant, 1/2009 - 05/2009
Department of Chemical Engineering,
University of Illinois at Chicago, Chicago, IL

- CHE 410 : Transport Phenomena

Teaching Assistant, 1/2008 - 05/2008
Department of Chemical Engineering,
University of Illinois at Chicago, Chicago, IL

- CHE 312 : Transport Phenomena II

PUBLICATIONS

- Kim, K. H., Rogelio H. Lopez, and Lewis E. Wedgewood, Stochastic Method Using Constraints for Modeling of Blood Rheology, *submitted to Physical Review E*
- Kim, K. H. and Lewis E. Wedgewood, Stochastic Method Using Constraints for Modeling of Blood Rheology: Part II, *in preparation*

CONFERENCES ATTENDED

- Presentation at the Society of Rheology (SOR) 86th Annual Meeting, Philadelphia, PA, 2014
- Presentation at 2014 Midwest Thermodynamics and Statistical Mechanics Conference Chicago, IL, 2014
- Poster Presentation at 5th Annual AIChE Midwest Regional Conference, Chicago, IL, 2014
- Presentation at the Society of Rheology (SOR) 84th Annual Meeting, Pasadena, CA, 2013
- Poster Presentation at the Annual Meeting of AIChE and ACS Great Lakes Chapter (GLCACS), Chicago, IL, 2013

PROFESSIONAL MEMBERSHIP

- Society of Rheology (SoR)
- American Institute of Chemical Engineers (AIChE)
- The Society of Women Engineers (SWE)
- Korean-American Scientists and Engineers Association (KSEA)
- Korean Institute of Chemical Engineers (KIChE)