ABSTRACT

Title of Document: FINITE-DISCRETE ELEMENT METHOD SIMULATIONS OF COLLIDING RED BLOOD CELLS

Benjamin William Warner, M.S., 2014

Directed By: Associate Professor Santiago Solares, Department of Mechanical Engineering

The implantation of artificial heart valves can lead to a large decline in red blood cell life. There has been much research in the last few years dedicated to understanding the cause of this decline. One theory states that collisions at large velocity can lead to spontaneous hemolysis which leads to the premature recycling of cells by the body. Currently, there is no suitable method for modeling the complex intersection interaction of blood cells in a computer code. The Finite-Discrete Element Method (FDEM) is a relatively new computer modeling technique that seeks to combine modeling of continuum-based deformability and discontinuum based motion and element interaction. This thesis utilizes FDEM to model the collision of erythrocytes with other erythrocytes. A method of approximating volume of arbitrary discrete element meshes is proposed and tested for general colliding bodies for accuracy. Red Blood cell simulations are presented with experimentally verifiable data to allow for validation of the model. Future steps are presented for further development of the

model for more specialized applications, such as sedimentation and resting contact. The volume-based FDEM method appears to recreate reasonable results for colliding deformable bodies.

FINITE-DISCRETE ELEMENT METHOD SIMULATIONS OF COLLIDING RED BLOOD CELLS

By

Benjamin Warner

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Masters of Science
2014

Advisory Committee:
Associate Professor Santiago Solares, Chair
Professor Hugh Bruck
Associate Professor Teng Li

Dedication

To my Father,
John Calvin Warner Jr.,
in loving memory.

Acknowledgements

I would like to thank my friends and family who have been so supportive in the pursuit of my research. Without their support, this work would not have come to fruition.

I would like to specially thank my advisor Dr. Santiago Solares who provided not only intellectual guidance, but emotional support as I worked through particularly difficult times. The gratitude I have for his unwavering support and tutelage is impossible to quantify. In addition, I would like to thank my research group for all that they have provided me over the last several years mentally and emotionally: Steve Oursler, Adam Kareem, Daniel Ebeling, Alan Wright, Hussein Ezzeldin, Babak Eslami, Enrique Lopez, Alfredo Diaz Gonzalez, and Sarice Barkley. The work environment fostered by these individuals has been nothing short of incredible.

I would like to give special recognition to Hussein Ezzeldin for his willingness to lend me knowledge and extended periods of time from his busy schedule for coding support.

I would like to thank the University of Maryland and all of its faculty and staff who have assisted me in becoming the engineer that I have become over the past 6 years.

I would like to acknowledge the financial support from the U.S. National Science Foundation (NSF) OCI-0904920 for the project discussed herein.

Table of Contents

List of Figures

Chapter 1: Motivation

Section 1.1: Artificial Heart Valves and Hemolysis

Heart disease has consistently been ranked as the leading cause of death in the United States of America for the past several years [1]. Because heart disease accounts for such a large number of deaths, researchers have focused on ways of improving heart health, such as the implantation of artificial and biological replacement heart valves (see Figure 1) when one's biological heart valve becomes too weak to work effectively.

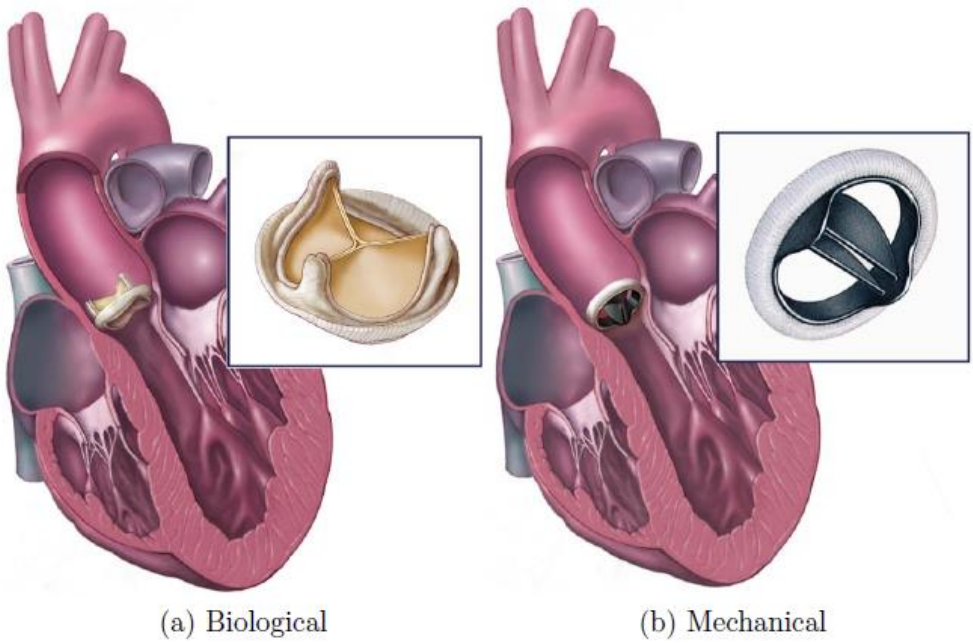

Figure 1: Image of two different types of replacement heart valves. Image taken from (http://www.mayoclinic.org/heart-valve-surgery/types.html).

The implantation of a replacement heart valve, however, leads to a sharp decline in the life of a red blood cell. A typical red blood cell has a lifespan of roughly 100-120 days. After implantation, the life of a red blood cell can drop by nearly 20% and the patient must take blood-thinning medication for the remainder of their life to prevent

blood clotting on the implanted device. The quality of life of the patient drops considerably due to the implantation of valve devices. Considerable research has been conducted to understand the change in blood flow patterns and mechanical reactions of the blood cells to these new flow patterns. Theories have been developed that will be discussed in Chapter 2 for quantifying blood damage with respect to stresses and strains. Research conducted by Yagi [2] established the presence pockets of instantaneous high-frequency shear stress downstream of an artificial heart valve shown in Figure 2.

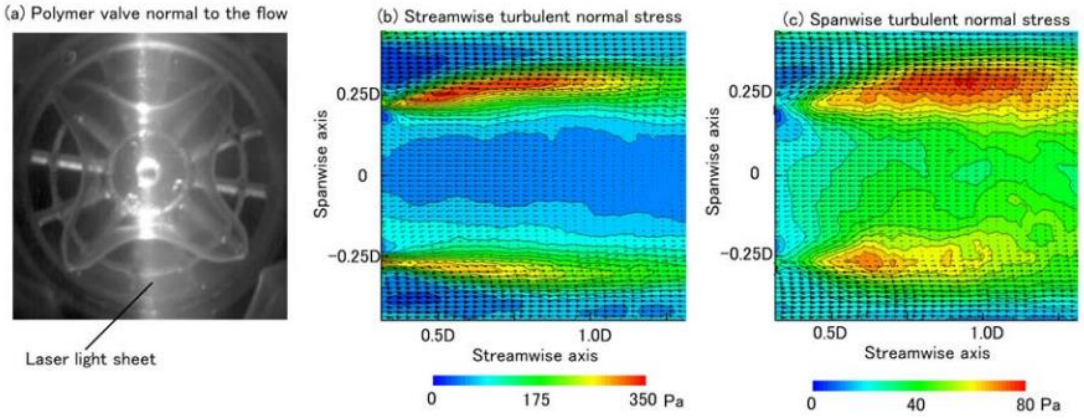

Figure 2: Turbulent normal stress fields downstream of an artificial heart valve. The flow was a fully developed 15 L/min flow. The pockets of high stress are not typically seen in this magnitude in biological heart valves. Image from [2].

Yagi also observed an entrainment effect that lead to these pockets having a noticeably higher density of red blood cells than typical flow patterns. Later work was conducted by the same group [3] that established the formation of vortices in the artery, downstream from the valve. An aortic sinus formation shown in Figure 3 is common in patients who have an artificial heart valve implantation.

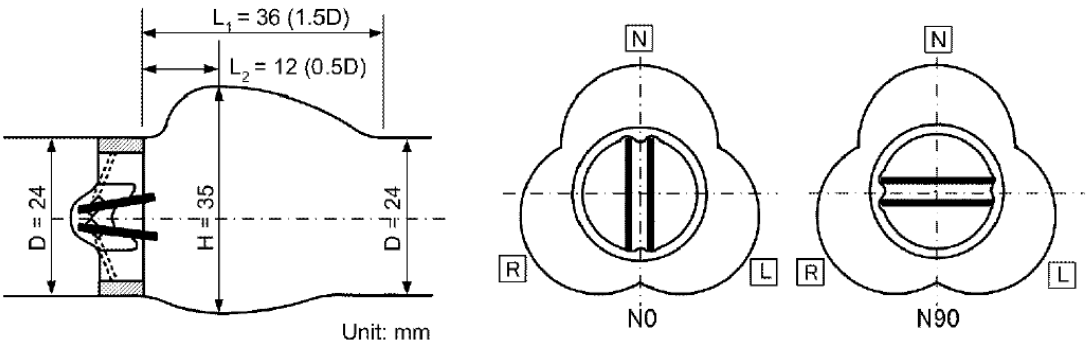

Figure 3: Aortic sinus used in experimental setup from [3]. A bulge such as this is often found in artificial heart valve patients, and is believed to be part of the cause of the shortened lives of red blood cells.

The aortic sinus found in these patients creates irregular flow patterns that subject red blood cells to high stress flow patterns, which are thought to be a leading cause for the hemolysis that shortens red blood cell lifespan. Yagi finds in his study that the so-called "instantaneous high-frequency stress fields" lead to pockets of vortices in which there is cell entrainment that serves to trap cells. This entrainment could lead to higher instances of cell collision when compared to a biological heart valve; therefore the collisions of red blood cells will be considered of great importance when examining the damage to red blood cells in the altered flow states created by an artificial heart valve. The following section will justify the importance of red blood cell collisions with respect to their role in shortening a cell's life span.

Section 1.2: Applicability to Scheme of Project

The red blood cell simulation code, upon arriving to this project, was still in early stages of development. The intention of this code is to be able to place millions of red blood cells into an artery model having an artificial heart valve and be able to simulate the interactions of the cell with the surrounding fluid, artificial heart valve,

and other red blood cells. It is proven that it is not correct to assume that there is a fluid boundary layer around red blood cells while flowing, so there is little preventing cells colliding in blood vessels [4]. Red blood cells in low Reynolds Number flow patterns have been shown to collide frequently and remain in contact [5]. This so-called rouleaux formation tends to occur more often in low flow patterns and was not of particular interest when considering the flow patterns of major arteries downstream of major arteries. These experiments do, however, establish the idea that a fluid boundary layer is not sufficient to prevent cell collisions and that collisions become more important as the cell ages.

The contact model becomes particularly interesting in accurately recreating the fluid flow of particles without a pre-defined path. The simulations that will most accurately capture flow phenomena are those in which the cells are allowed to interact with each other and the fluid particles, with the resulting flow patterns not being pre-determined. This may allow for the observation of previously unforeseen occurrences that can give a more complete picture of the hemolysis problem that we are seeking to describe. The simulations run by Ezzeldin [6] with the red blood cell code simulate the red blood cell as it flows along prescribed flow paths. The cell only sees fluid and viscous forces on its membrane surface, which may not necessarily fully describe the stresses and strains felt by the red blood cell.

It has been theorized by Yagi et al. [7, 8] that cell collisions can cause instantaneous ejection of hemoglobin from the cell. Red blood cells, under high-speed flow

conditions, undergo strain hardening which makes the membrane brittle despite being a fluid-filled sac. When a brittle red blood cell contacts a wall, it undergoes buckling as seen in Figure 4 below. The buckling puts an extreme strain on the cell membrane which causes an instantaneous ejection of hemoglobin. This loss of hemoglobin has been theorized to play an active role in determining when the body recycles older red blood cells at the end of their life.

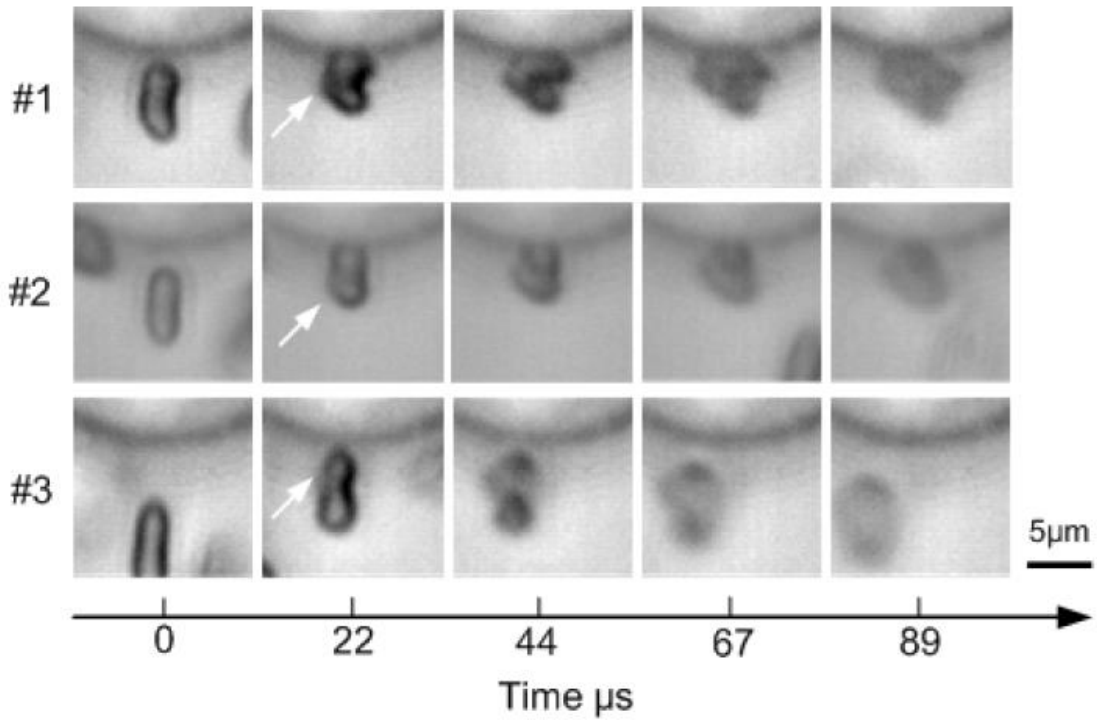

Figure 4: High-speed video of red blood cell colliding with a Y-shaped junction. The arrow marks a hazy ejection from the cell that is theorized to be hemoglobin. Image from [7].

The question then becomes at what frequency red blood cells can collide during flow downstream from a heart valve and to what extent an artificial heart valve can impact this collision frequency. It would make sense to assume that there is a boundary layer between all erythrocytes that prevents collisions. However, Trowbridge [4] develops a physical description of arterial blood flow and concludes that it is not appropriate to think of blood flow as a collection of red blood cells surrounded by a continuum

fluid. Such an approximation is inappropriate because the cells contribute to the flow patterns that are typically described as general blood flow. As such, the cells and plasma "should be considered as continua which combine to produce the fluid referred to as blood". In Trowbridge's experimentation, collisions with arterial walls are seen, which dispels the notion that boundary layers dominate red blood cell interaction. Trowbridge further explains that macroscale observation of a flow can lead to an incorrect characterization of flow patterns because of an averaging effect due to improper visualization equipment.

Thus, by establishing that blood should be treated as a fluid instead of a fluid with solid particles suspended in the flow, it is possible to state that red blood cells are able to collide with each other and artery walls. This conclusion will be the basis of the simulations conducted in the following chapters.

Chapter 2: Introduction

Section 2.1: Goals of this Project

The scope of this project is to create a Finite-Discrete Element Method code that can be implemented into a pre-existing FLASH code capable of simulating large numbers of red blood cells in order to track and analyze damage to the red blood cells as they flow through an artificial heart valve. The methods used in creating such a code will be detailed through the following sections. After the components of the code are reviewed and shown to be appropriate for their application, simulation data will be presented and analyzed for plausibility.

As it stands, there is no experimental data that presents collisions between red blood cells in a controlled environment. As such, a range of data was collected and presented to allow for easier fitting of experimental data when such experiments are conducted in the future. Because this code was developed to be a part of a larger pre-existing code, special sections are included to discuss efficiency and how the contact model fits into the Fortran and FLASH architecture.

Section 2.2: Coarse-graining

A typical red blood cell has roughly 24,000 junction complexes that define the spectrin network that acts as a skeleton for the cell membrane [9]. The spectrin network gives the red blood cell its rigidity and gives rise to many of the properties that are desirable in simulating its dynamic response. As such, in order to obtain an accurate computer model of a red blood cell, it would make sense that the model should contain 24,000 nodes, effectively modeling the junction complexes. However,

the computation time involved in simulating this number of nodes is a significant hurdle in expanding the model to large-scale simulations with millions of red blood cells. This becomes especially evident in looking at the simulation time of the collision module even for a coarse-grained cell. As will be explained in Section 4.2, simulation time increases by an order of magnitude after inclusion of a collision module between just two coarse-grained red blood cells.

Thus, it is desirable to reduce the number of points required to fully define the surface of the red blood cell which will significantly reduce the computation time for simulations. In order to accomplish this coarse-graining, the method proposed by Pivkin [9] is adopted. Different parameters used in the model are weighted appropriately to come to a new model with fewer nodes. Geometric arguments based on the change in element area can be used to adjust the equilibrium length, L, of nodal links:

$$L_0^c = L_0^f \sqrt{\frac{N^f}{N^c}} \tag{2.1}$$

The c and f superscripts correspond to the parameters of the red blood cell when the mesh is fine (N^f=23867) and after it has been coarse grained (N^c<23867). Similar arguments can be made for the spontaneous angle, θ, and persistence length, p, as explained in the following equations:

$$\theta_0^c = \theta_0^f \frac{L_0^c}{L_0^f} \qquad p^c = p^f \frac{L_0^f}{L_0^c} \tag{2.2, 2.3}$$

Note that the persistence length is converted using the inverse value of the ratios of the square of the number of nodes. This relationship arises from the desire to keep the shear modulus of the cell membrane constant:

$$\mu_0 = \frac{\sqrt{3}k_B T}{4pL_{max}x}\left(\frac{3}{3(1-x)^2} - \frac{3}{4} + 4x + \frac{x}{2(1-x)^3}\right) \tag{2.4}$$

In order to keep the shear modulus constant regardless of the level of coarse-graining, p must be decreased by the same magnitude that L_{max} in increased. Further detail on all of these terms is found in the following section, but note that because x is a ratio of lengths, its value does not affect the shear modulus.

Pivkin tested various levels of coarse-graining using this simple method by simulating a stretching test and comparing it to a fine-mesh model. The axial and transverse diameters were measured at the different levels and plotted collectively to measure the feasibility of the model with the results shown in Figure 5.

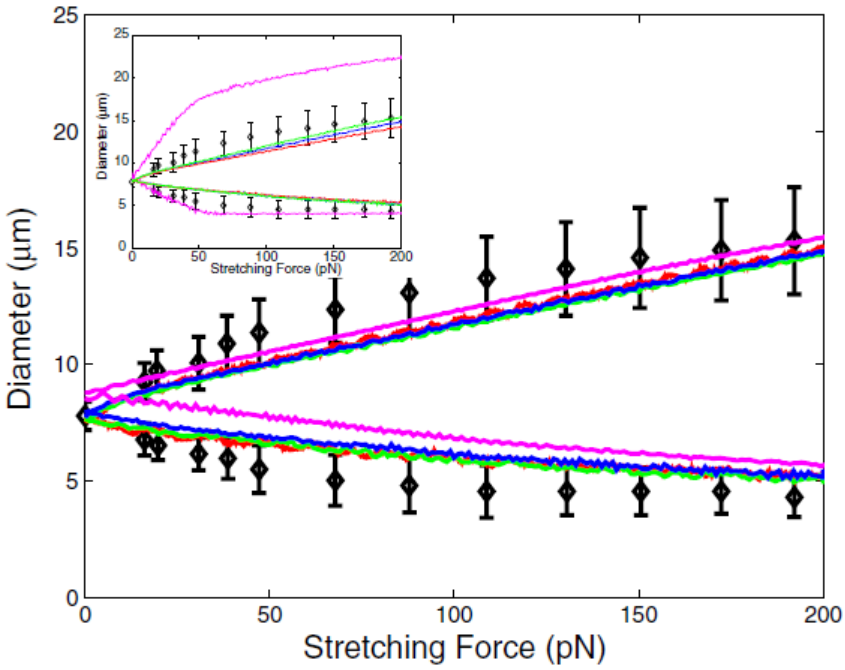

Figure 5: Axial and transverse diameters of the red blood cell from various stretching forces. The blue line is N=23867 nodes, red line is N=5000 nodes, green line is N=500 nodes, magenta line is N=100 nodes. Figure from reference [9].

The results were also visualized to see if the same geometric phenomena from a fine-meshed model could be recreated with a coarse-grained model, as per Figure 6.

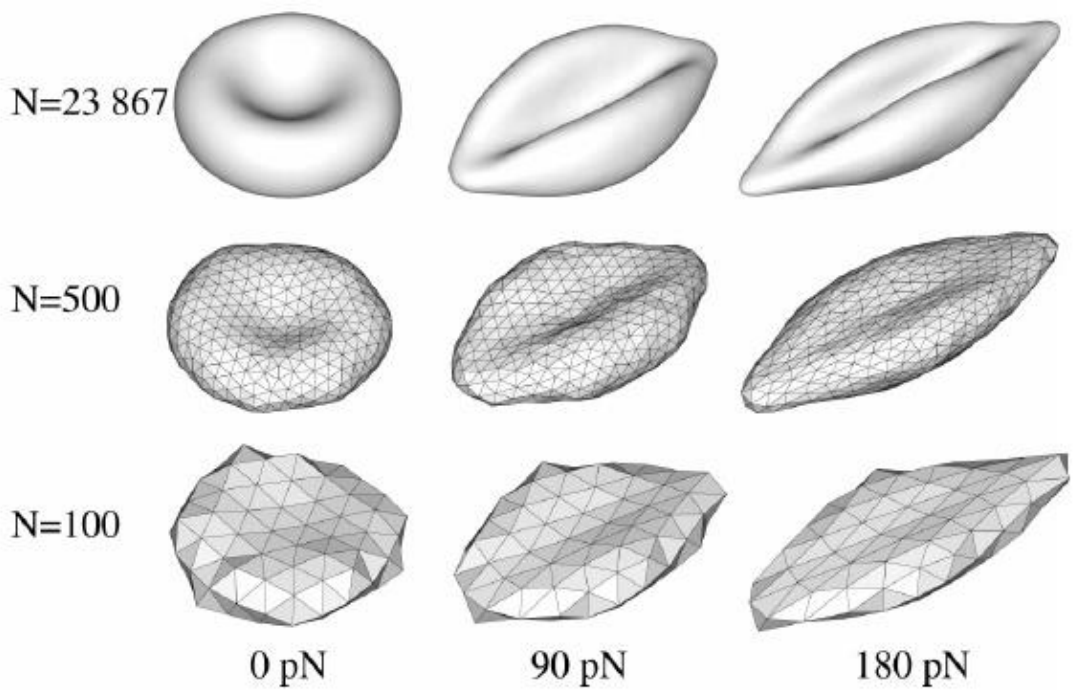

Figure 6: Stretching simulation visualization at different levels of coarse-graining related to the fine mesh over several forces. The general features from the stretching can be seen at even the highest level of coarse-graining, though the features become much less distinguishable beyond a certain point. Figure from reference [9].

Pivkin summarized his findings in concluding that the number of nodes should be restricted to greater than 100 in order to represent deformation mechanics. Fedosov [10] theorized that the model needs at least 500 nodes to accurately portray a red blood cell.

Fedosov [10, 11] later argued that pre-existing methods did not accurately estimate the membrane's Young's modulus. He argues that a model can be created that does not depend on the initial triangulation by introducing a stress-free model. The following section will detail the model that is proposed by Fedosov to alleviate the problems introduced by the Pivkin coarse-grained model.

Section 2.3: Fedosov Model

Fedosov [10, 11] developed a method of modeling the forces experienced on the membrane of a coarse-grained red blood cell that builds upon the method proposed by Dischер [12]. Fedosov first defines the free energy of the system as consisting of four individual energy terms that may be treated separately:

$$V_{tot} = V_{in-plane} + V_{bending} + V_{area} + V_{volume} \tag{2.5}$$

Figure 7: 2-point interaction between two adjacent nodes.

The $V_{in\text{-}plane}$ is a 2-point interaction (Figure 7) between a node and its neighbors that share a bond. For the purposes of this thesis and the simulations run within, the attractive spring force that will be used is the worm-like chain model. As such, the attractive potential that will govern 2-point interactions will be:

$$U_{WLC} = \frac{k_B * T * l_{max}}{4p} * \frac{3x^2 - 2x^3}{1 - x} \tag{2.6}$$

Where p is the persistence length, l_{max} is the limit placed on the maximum extension of any given 2-point interaction, and x is the fraction defined by $\frac{l}{l_{max}}$. The length of any link is limited to be less than l_{max} such that the denominator of the above equation approaches zero as the length approaches l_{max}. Thus, as the link length approaches l_{max}, the attractive potential approaches infinity. Because of the restriction that x is less than one, the potential above is always positive so a repulsive potential is necessary to counteract the worm-like chain potential. Again, for the purposes of this

thesis, the repulsive force will be defined only as a power model, though other models were discussed by Fedosov:

$$f_{POW} = \frac{k_p}{l^m} \tag{2.7}$$

The exponent, *m*, was chosen to be two, while the coefficient k_p arises from equating the above equation to the worm-like chain at equilibrium length.

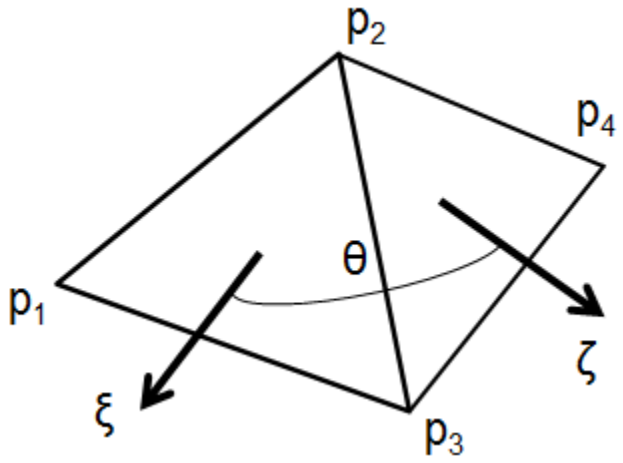

Figure 8: 4-point interaction between elements that share two common nodes.

$V_{bending}$ is a 4-point interaction (see Figure 8) between two elements that share a common link. An equilibrium angle is set based on the desired geometry such that the potential energy can be defined as:

$$V_{bending} = \sum_{i=1\ldots N_v} k_b(1 - \cos(\theta - \theta_i)) \tag{2.8}$$

Where N_v=3*N_{Nodes}-6, k_b is the bending constant that is derived from the bending stiffness, θ is the equilibrium angle, and θ_i is the instantaneous angle formed between the normals of two adjacent elements.

The V_{area} term is defined by a harmonic of two separate terms:

$$V_{area} = \frac{k_a(A - A_0^{tot})^2}{2A_0^{tot}} + \sum_{i=1\ldots N_t} \frac{k_d(A_i - A_0)^2}{2A_0} \tag{2.9}$$

A represents the total surface area of the red blood cell. A_0^{tot} is the equilibrated surface area of the red blood cell and can be thought of as the zero position of the potential, such that when the total area equals this equilibrated area, there is no force due to surface area. The A_i and A_0 terms are the individual elemental areas and the equilibrium area for individual elements, respectively. k_a and k_d are the area constants for each of the above terms. The sum of the k_a and k_d terms is restricted to be 5000. In the simulations run in this thesis, the k_d term is actually set to be zero, so the second term is dropped from the simulation completely.

The final term in Equation 1.10 is a simple harmonic for the volume potential:

$$V_{volume} = \frac{k_v(V - V_0^{tot})^2}{2V_0^{tot}} \tag{2.10}$$

V represents the instantaneous volume of the red blood cell. V_0^{tot} is the equilibrium volume of a red blood cell that is equal to the average measured volume of a red blood cell. The k_v term is the volume spring constant which takes the value of 5000 in the simulations.

It is at this point that the stress-free model will be briefly discussed. If only one equilibrium length is defined for the worm-like chain/power model spring potential, some geometric anomalies may arise that lead to incorrect results. To be able to avoid these anomalies, a stress-free model was proposed that tracked initial equilibrium link lengths. After the points have been appropriately placed on the

analytic surface, the program can then save the equilibrium lengths of all of the links separately. In addition to a new equilibrium length, updated k_p values are created in order to appropriately balance the potential at equilibrium length as discussed earlier in this section. The other three terms remain unchanged because they depend on higher level geometry that is independent of equilibrium spring length.

Once the various spring energy potentials have been defined, the force that results from changing the potential can be derived from the work-energy relationship in order to govern the motion of the red blood cell:

$$f_i = -\frac{dV}{dx_i}, \qquad i = 1 \dots N \tag{2.11}$$

N is the dimensionality of the system that is being examined.

We will now go through the steps in developing the force equations from the energy potentials. We begin with the 2-point interaction:

$$f_{in-plane} = \frac{dV_{in-plane}}{dl} = \frac{dU_{WLC}}{dl} + f_{POW} \tag{2.12}$$

$$\begin{aligned} \frac{dU_{WLC}}{dl} = f_{WLC} &= -\frac{k_B T l_{max}}{4p}\left[\frac{1}{l_{max}}\left(4x + \frac{1}{(x-1)^2} - 1\right)\right] \\ &= -\frac{k_B T}{p}\left(x + \frac{1}{4(x-1)^2} - \frac{1}{4}\right) \end{aligned} \tag{2.13}$$

The power repulsion model previously discussed was introduced as a force, so no differentiation is required. The total in-plane force is therefore given as:

$$f_{in-plane} = -\frac{k_B T}{p}\left(x + \frac{1}{4(x-1)^2} - \frac{1}{4}\right) + \frac{k_p}{l^m} \tag{2.14}$$

The force above is a magnitude of the entire in-plane force so an appropriate direction must be given to the force. The in-plane interaction is a 2-point interaction, so the force simply acts along a unit vector between the two interacting points.

For the area potential, a similar analysis can be conducted:

$$f_{area} = -\frac{dV_{area}}{ds_i} = -\frac{k_a(A - A_0^{tot})}{A_0^{tot}}\frac{dA}{ds_i} \tag{2.15}$$

Note that the first term in the equation above is a constant among all triangles. The second term can be broken into a summation that takes the form:

$$\frac{dA}{ds_i} = \sum_{j=1\ldots N_t} \frac{dA_j}{ds_i} \tag{2.16}$$

The algebra used to arrive at the usable form of this equation is found in [13], but the final result for the force due to area potential is:

$$f_{area_i} = -\frac{k_a(A - A_0^{tot})}{4A_0^{tot} * A_m} * (\xi \times a_{kj}) \tag{2.17}$$

In this equation, ξ is the normal of element m, A_m is the area of element m, and the indices *i*, *j*, and *k* denote the positive permutations of <1,2,3>, such that if *i*=2, then *j*=3, and *k*=1. The a_{kj} term is a vector that gives the rate of change of the vector between nodes *j* and *k*. Its use is explained further in Section 4.2.1. Note that the cross product gives a vector, so the f_{area} term actually has the dimensions of ξ, indicating an N-dimensional force. Another important note is that each node will experience a force from each of the elements that it is contained within, so equation 2.17 does not represent the total force due to surface area conservation on a node, but rather a portion of the total.

Analogous calculations from the area potential can be used to calculate the force due to volume conservation. The force takes the form of the equation 1.18 as:

$$f_{area} = -\frac{dV_{volume}}{ds_i} = -\frac{k_v(V - V_0^{tot})}{V_0^{tot}}\frac{dV}{ds_i} \tag{2.18}$$

Similarly, the rightmost term can be divided into a summation:

$$\frac{dV}{ds_i} = \sum_{j=1\ldots N_t} \frac{dV_j}{ds_i} \tag{2.19}$$

The individual volumes can be calculated using the volume of parallelepipeds, $V_j = \frac{1}{6}(\vec{\xi}^k \cdot \vec{t}_c^k)$. Taking the derivative as shown above will then yield:

$$f_{volume_i} = -\frac{k_v(V - V_0^{tot})}{6V_0^{tot}} * \left(\frac{\xi}{3} + t_c \times a_{kj}\right) \tag{2.20}$$

As was the case in the area force equation, *i*, *j*, and *k* denote the positive permutations of <1,2,3>. All of the other symbols are consistent with how they have been previously explained. The result of this force calculation is again an N-dimensional vector.

The final force to be derived is the four-point interaction that arises from $V_{bending}$. For brevity, the final result will be given as derived in [13]:

$$f_{bending_1} = b_{11}(\xi \times a_{32}) + b_{12}(\zeta \times a_{32}) \tag{2.21}$$

$$f_{bending_2} = b_{11}(\xi \times a_{13}) + b_{12}(\xi \times a_{34} + \zeta \times a_{13}) + b_{22}(\zeta \times a_{34}) \tag{2.22}$$

$$f_{bending_3} = b_{11}(\xi \times a_{21}) + b_{12}(\xi \times a_{42} + \zeta \times a_{21}) + b_{22}(\zeta \times a_{42}) \tag{2.23}$$

$$f_{bending_4} = b_{12}(\xi \times a_{23}) + b_{22}(\zeta \times a_{23}) \tag{2.24}$$

Where ζ is the normal of the second element, and $b_{11} = -\frac{\beta_b \cos\theta}{|\xi|^2}$, $b_{12} = \frac{\beta_b}{(|\xi||\zeta|)}$, $b_{22} = -\frac{\beta_b \cos\theta}{|\zeta|^2}$, and $\beta_b = \frac{k_b(\sin\theta\cos\theta_0 - \cos\theta\sin\theta_0)}{\sqrt{1-\cos^2\theta}}$.

All of the internal forces due to energy potentials that are required for a physically accurate red blood cell model have been fully defined.

There may be, however, the need for inclusion of a damping force to allow the body to minimize its total energy upon deformation. Espanol [14] created a fluid particle model that applies damping forces at individual nodes. In his work, he cites the existence of four main forces in a fluid model:

$$F_{ij} = F_{ij}^C + F_{ij}^T + F_{ij}^R + \tilde{F}_{ij} \tag{2.25}$$

Where F_{ij}^C is the conservative force on a node, F_{ij}^T is a frictional force that is dependent on relative velocities of adjacent nodes, F_{ij}^R is a dissipative force due to rotation, and the final term contains random forces. The conservative force was not dealt with in these simulations as they were run in an assumed vacuum, absent of fluid particles and their effects. Likewise, because the nodes are not assumed to have a rotational orientation as they are generalized to be volumeless points, there is no need to consider rotational dissipation forces. There were two dissipation models that are built in to the pre-existing code based on the remaining two non-zero forces from the above equation. The model could be selected in the input.dat file as explained in Section 4.1.

The first viscous model involved the use of random forces through the use of Weiner matrix increments. This model was not used for the simulations presented in this thesis. For further detail on the random force application, refer to [14, 6]. The model that was used for the simulations in Chapter 5 is derived from the frictional force from the relative velocities of linked nodes. The form of this force is:

$$F_{ij} = -\gamma v_{ij} \tag{2.26}$$

In the above equation, γ is a frictional coefficient that is selected to be 0.5 for all simulations in this thesis. Note that this equation creates equal and opposite forces on any two nodes that are interacting with each other. This satisfies the conservation of momentum that is necessary for physically accurate Molecular Dynamics simulations. This concludes the list of forces that are experienced during the simulations presented herein. Fluid forces, though of great importance and interest, were ignored to allow for the analysis of collisions in vacuum to isolate the collision phenomena from outside influences.

Section 2.4: Non-Dimensional Simulations

For the purposes of this thesis, the simulations and data presented were run in a non-dimensional form. There were several reasons for performing this scheme. The most prominent reason pertains to the scale on which the experiments would typically be performed. Though Fortran is capable of handling numbers on the order of 10^{-46}, it is possible for some values to fall below that value. In particular, the Worm-like Chain model led to experiencing several issues with calculating NaN and Inf errors. These errors arise from Fortran's inability to handle extremely large and extremely small values. If a number in Fortran exceeds the maximum value that can be handled, it is

considered infinite and will remain infinite regardless of the computations performed on it. NaN errors came from Fortran's inability to handle very small numbers. If a number becomes smaller than the smallest value Fortran can handle, it rounds the value to zero. It is possible for certain values in the simulation to become smaller than this minimum and then used as denominators. NaN is the error that results from division by zero. Thus, in order to prevent errors related to values outside of the range that Fortran can handle, the values were non-dimensionalized which changed the scale to be closer to integers when talking about length.

Second, because the data being presented here will deal with largely qualitative analyses and dimensionless quantities, the data will be kept dimensionless. The scheme for developing this non-dimensional simulation was adopted from Fedosov [11] and will be described briefly here. First, a length scale needs to be developed based on the number of nodes in the model as follows.

$$r^M = \frac{D_0^P}{D_0^M} \tag{1.27}$$

Where the *M* and *P* superscripts correspond to model and physical parameters. The *D* variable is the average diameter of the red blood cell. For the physical model, $D_0^P = 7.82\mu m$ and the simulation D_0^M=8.25. To non-dimensionalize the energy, the Young's modulus was used as the scaling parameter. Symbolically,

$$\frac{Y^M (k_B T)^M}{(r^M)^2} = Y^P (k_B T)^P \tag{1.28}$$

Isolating the model energy term and substituting Equation 1.28 above,

$$(k_B T)^M = \frac{Y^P}{Y^M}\left(\frac{D_0^P}{D_0^M}\right)^2 (k_B T)^P \tag{1.29}$$

The Y^P value is selected to be 18.9 μN/m while the Y^M parameter is chosen to be 392.453. The force also needs to be non-dimensionalized in a similar manner

$$F^M = \frac{Y^P}{Y^M}\frac{D_0^P}{D_0^M} F^P \tag{1.30}$$

Lastly, time needs to be non-dimensionalized. Time will be the only parameter that is converted back to physical terms to allow for a more simple understanding of the visualization presented in the data section. To non-dimensionalize time,

$$\tau = \left(\frac{D_0^P}{D_0^M}\frac{Y^M}{Y^P}\frac{\eta^P}{\eta^M}\right)^\alpha t \tag{1.31}$$

In the above equation, η is the viscosity of the model and physical system, respectively. For these simulations, η^P is set to 0.022 Pa•s while the model value is set to be 0.022 as well. α is set to be 0.85.

Section 2.5: Blood Damage Quantification

Because the life of red blood cells drops so dramatically upon the implantation of an artificial heart valve, researchers have attempted to quantify the damage a red blood cell incurs based on its flow patterns. There are two main methods that have been hypothesized to contribute to red blood cell damage.

The first quantity to be discussed here relates to the amount of stress felt by the links and how that leads to the accumulation of hemolysis. One of the most popular stress-based models was proposed by Giersiepen [15] which relates shear stress and the

amount of time exposure for that shear stress to the amount of hemolysis and hemolysis rate. Hemolysis is largely affected by not only the magnitude of shear stress, but also the exposure time to that shear stress, making it important that proposed models incorporate both factors. A list of characteristics that a hemolysis model should contain can be found in [16], but of particular importance to the collision model presented in this thesis is the idea that physiological blood flow conditions may affect hemolysis significantly, so the hemolysis model should be modified to account for red blood cells contacting surfaces. The model proposed by Giersiepen relates the hemolysis and hemolysis rate to the shear stresses and time through a power law:

$$\frac{\Delta Hb}{Hb} = 3.62e10^{-7} \cdot \tau^{2.416} \cdot \Delta t^{0.785} \cdot 100\% \tag{1.32}$$

$$\frac{d}{dt}\frac{\Delta Hb}{Hb} = 2.8417e10^{-7} \cdot \tau^{2.416} \cdot \Delta t^{-0.215} \tag{1.33}$$

τ is the shear stress experienced over the course of the time step Δt. The apparent weaknesses of this model are due to the inability to account for varying shear stresses as would be expected from a typical red blood cell flow pattern and the inability to deal with accumulated damage. It has been found that the properties of a blood cell change as the cell ages, so there is a necessity to track cell damage throughout its life which the above equations are incapable of doing. Grigioni then proposed a model that attempted to incorporate both of these properties.

$$BDI = \sum_{i=1\ldots N} C \cdot a \left[\sum_{j=1\ldots i} \tau(t_j)^{\frac{b}{a}} \Delta t_j + D(t_0) \right]^{a-1} \cdot \tau(t_i)^{\frac{b}{a}} \cdot \Delta t_i \tag{1.34}$$

The *C*, *a*, and *b* are constants, *N* is the number of time steps for which the blood damage index is being measured, and *D* is the damage of the red blood cell at time t_0.

The second method for calculating red blood cell damage is through strain-based hemolysis modeling. The strain-based hemolysis attempts to quantify the deformation of the red blood cell through the use of an area morphology tensor [17]. Calculating the eigenvalues of a red blood cell gives the most and least likely orientation for deformation and the eigenvalues represent the proportion of interfacial area. The tensor changes throughout the simulation of a red blood cell along a pathline, which allows for the creation of a distortion parameter defined as:

$$\phi = \frac{L - B}{L + B} \tag{1.35}$$

L and *B* are the lengths of the radius of the red blood cells in the principal directions of the morphology tensor, otherwise calculated as the eigenvalues of the area morphology tensor.

Ezzeldin [6] proposed a method of quantifying the shear stresses on the discretized blood cell through the following means:

$$\tau_{\alpha\beta}|_{springs} = -\frac{1}{S}\left[\frac{1}{2}\sum_{i=1\ldots cn}\frac{f(r_i)}{r_i}r_i^{\alpha}r_i^{\beta}\right] \tag{1.36}$$

Where $S = \frac{cn}{3}\sum_i^{cn}\frac{A_i}{cn}$ is the representative area element of the node of interest. f is the force between two nodes, r_i are the coordinates of connected vertices, *cn* is the number of vertices neighboring a particular node, and α and β are x or y. The second shear stress value due to element area was defined as:

$$\tau_{\alpha\beta}|_{area} = -\left[\frac{1}{3}\sum_{i=1}^{cn}\frac{k_a(A_{oi} - A_i)}{A_{oi}}\right]\delta_{\alpha\beta} \tag{1.37}$$

k_a is the area coefficient defined earlier in the chapter, and A_{oi} and A_i are the initial and current area of a given element, respectively. From these values, Ezzeldin was able to apply the methods from previous work suggested by Evans [18]. In the work presented by Evans, a strong correlation between loading rate and membrane failure was found. Evans is able to relate defect propagation to load rate using cavitation theory of a small hole in two-dimensional film. By finding the discretized loading rate applied to the cell links, a relationship to membrane failure was determined.

This thesis will present a method of dealing with red blood cell interaction that will allow for the expansion of current hemolysis models to account for collisions and their possible impact on hemolysis.

Chapter 3: The Discrete Element Method

Section 3.1: History

Molecular Dynamics is a tool developed in the 1950s originally based around the Monte Carlo simulation technique [19]. Such techniques were used to determine dynamics and physical conformations of small, granular systems [20, 21]. Later, Molecular Dynamics simulations were conducted, and the first simulation of a real liquid was reported by [22]. MD is used widely to study liquids due to liquid flow's lack of theoretical development [23]. However, the MD method is also used to study physical phenomena in both gas and solid particles [24]. Rapaport lists several physical phenomena that have been discovered with the use of MD simulations, some of which include diffusion, vibration collective behavior (as in Figure 9), laminar fluid flow, and the structure and dynamics of proteins.

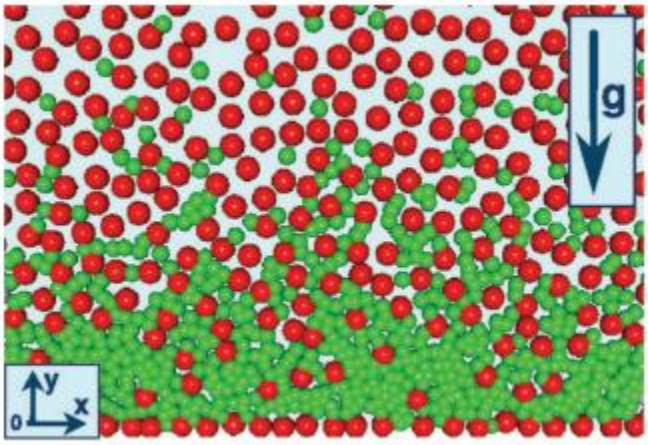

Figure 9: Simulation image of the so-called "Brazil nut effect" in which smaller particles tend to fall towards the bottom of the simulation area. Image from [25].

Molecular dynamics is a very versatile technique effective in modeling particles and their interactions, making it particularly well-suited to deal with the collisions of red blood cells as will be discussed in this paper.

Section 3.2: 2-D Applications

As computers have become faster, the complexity of simulations has increased. However, the application of the discrete element method to three-dimensional problems has only been a recent development. For a long period of time, the simulations were restricted to small two-dimensional spaces with few particles. Still, such simulations could effectively recreate fluid phenomena such as the vortices seen in gas particles.

More applicable to the simulations presented later in this thesis, however, are composite particles. These composite particles are composed of several different particles, which allows for the creation of more complex geometry (see Figure 10) without significantly altering the simple Discrete Element Method codes. These particles can even be bonded together with soft bonds allowing for deformation simulations [26].

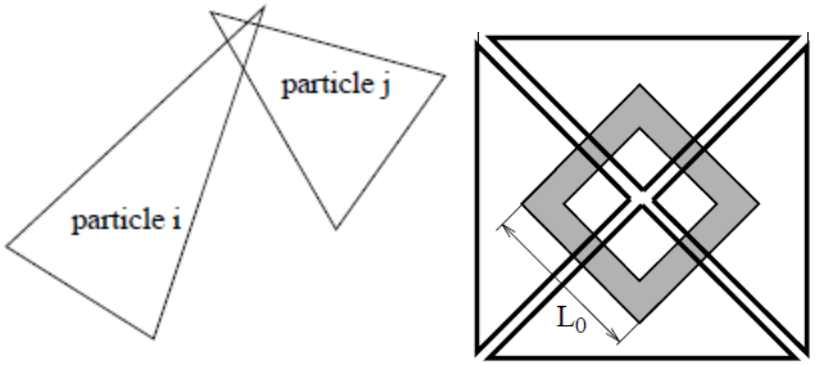

Figure 10: Examples of composite particles used in more complex 2-D DEM simulations. Images from [27] [26].

While each method works well in its respective application, we are seeking a method that treats the particles as soft composite bodies. That is, the body is comprised of slightly penetrable spheres or particles, with links that also allow for penetration (Figure 11).

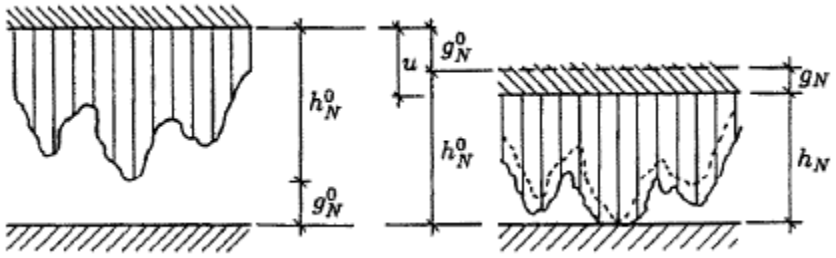

Figure 11: Discrete Element formulation of composite surface colliding with rigid surface. Image taken from [28].

In most two-dimensional discrete element applications, depth of penetration or area of collision are used to determine the force of repulsion and several approaches for determining this area have been proposed [29]. In addition, the type of contact, such

as vertex/face and edge/edge, is tracked which changes the types of force (see Figure 12) being applied [30]. This type of extensive bookkeeping is undesirable in large-scale simulations, so different methods were sought that were more suitable to the types of simulations proposed.

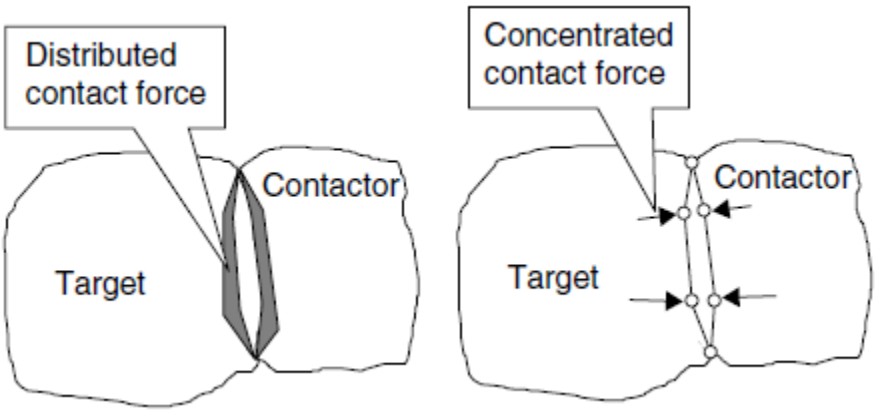

Figure 12: Contact force application that considers the body as a continuous surface (left) and a discretized surface (right). The rightmost image removes the extensive bookkeeping necessary in most applications. Image taken from [31].

Of particular note in the two-dimensional simulations were those proposed by Tsubota et al. [32] in which a blood vessel was simulated in two dimensions. Tsubota et al. make use of the so-called particle method where the surfaces of each blood cell are comprised of impenetrable circles as seen in Figure 13. This composite surface was allowed to flow freely in a fluid that was given a velocity and the red blood cell's deformation, shape, and velocity could be easily tracked.

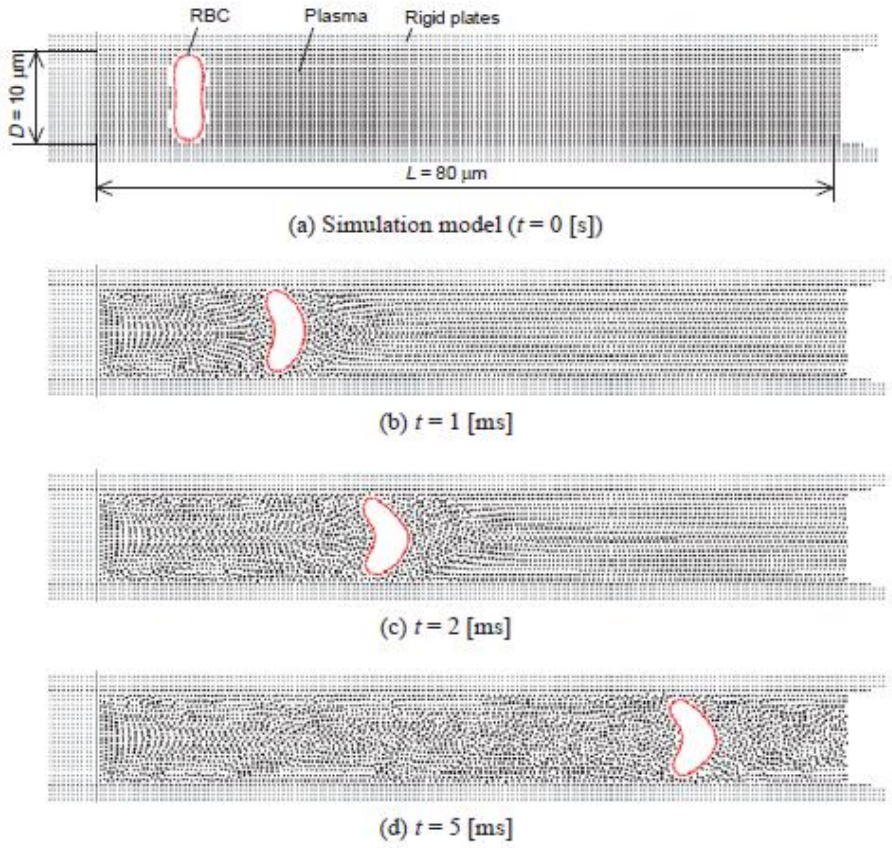

Figure 13: Particle method simulation of red blood cell flow through a blood vessel. Image taken from [32].

Although this method appears to work well for this specific application, the impenetrability condition is not easily implemented into the pre-existing red blood cell simulations used in this thesis. The full reason for avoiding the impenetrability condition is related to the time step integration technique used which is described in Section 4.1.3.

Section 3.3: 3-D Applications

Although most three-dimensional applications began as spheres for their relative simplicity for tracking their locations and detecting potential collisions, the discrete element method is suitable for many different types of particles, many of which are

not rigid. Because our simulation does not have regular geometry, methods for treating irregular geometry were examined for applicability. Several works were proposed where a particle was actually a composite of many smaller spheres that were rigidly connected. This allowed still for easy collision detection while achieving complex geometry. The composite method is widely used and is even available as open source software as in HoomD Blue [33]. The composite sphere bodies, however, were too simplified to be of use when dealing with a body that is only defined by surface points where collision detection is not straightforward.

Further methods were developed that dealt with defining the body as a network of links that allowed for simulation of any three-dimensional geometry with any level of mesh within reason. These methods ranged from the application of a "no penetration" condition [34] (see Figure 14) to allowing penetration where the force is directly proportional to the linear penetration distance [35] (see Figure 15).

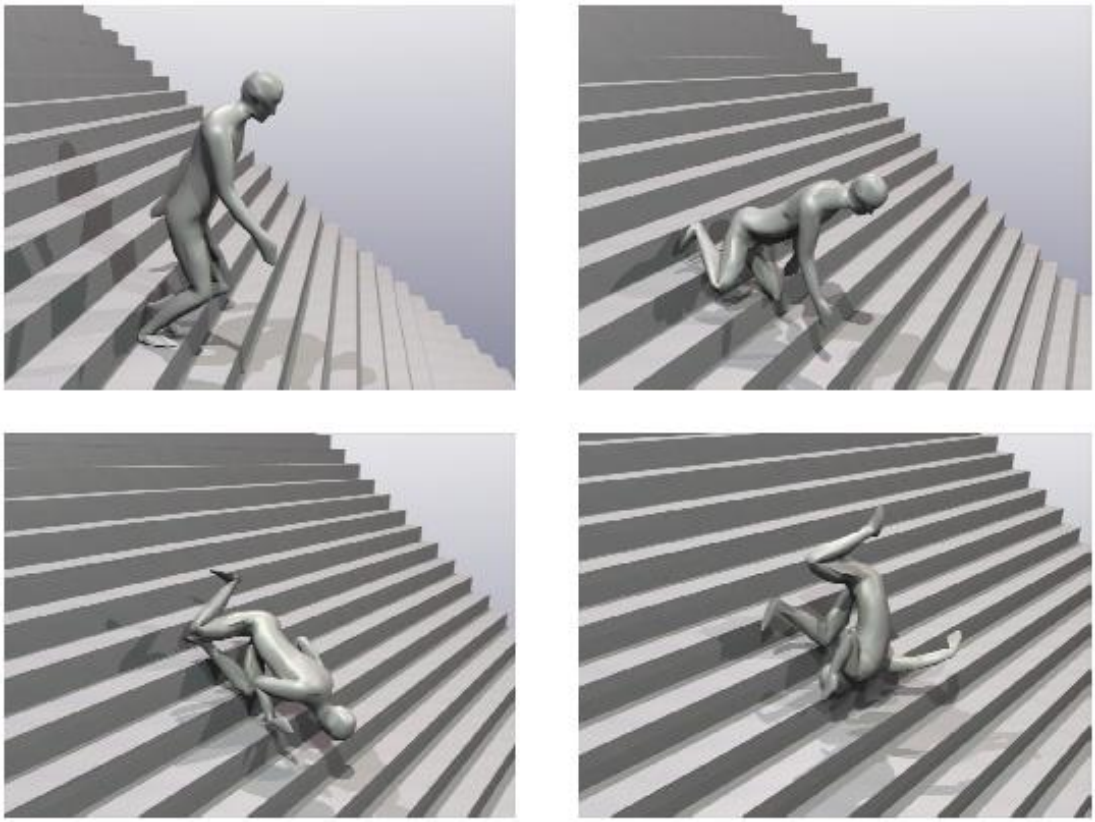

Figure 14: Demonstration of a three-dimensional simulation of a meshed human body with the no penetration condition. Image taken from [34].

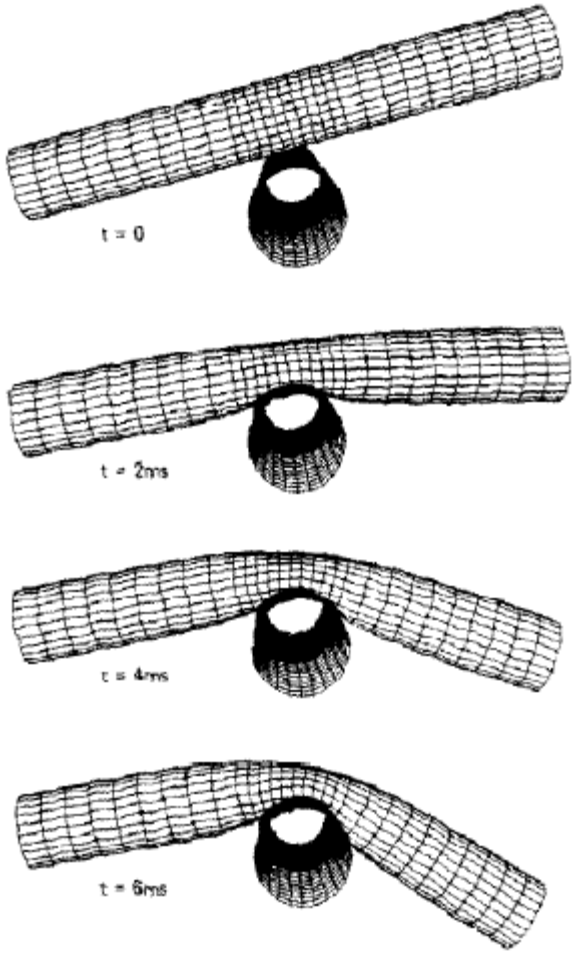

Figure 15: Three-dimensional simulation of two discretized cylinders in the penalty method in which penetration is allowed. Image taken from [35].

The reasons for choosing a particular method of discrete element simulations are explained in the respective sections in which the red blood cell collision module is developed. One of the most applicable simulations found while conducting this research was from Frenning [36, 37]. In Frenning's work, powder particles used in pharmaceutical applications are compacted. The particles are highly deformable and defined by nodes on the surface of the particles. In this sense, the particles used here closely resemble the red blood cells used in this thesis. Frenning uses a combined finite-discrete element model that treats each particle as a continuous body capable of

deforming coupled with the movements and interactions from the standard discrete element method as shown in Figure 16 and Figure 17.

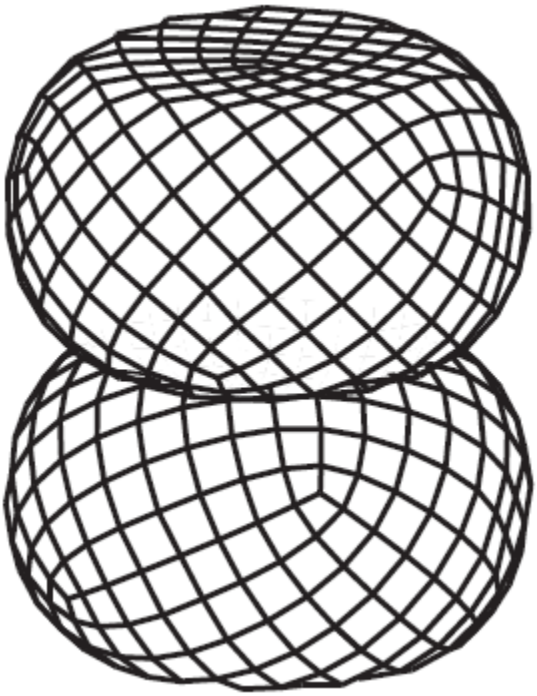

Figure 16: Deformed state of discrete element simulations proposed by Frenning. Image taken from [36].

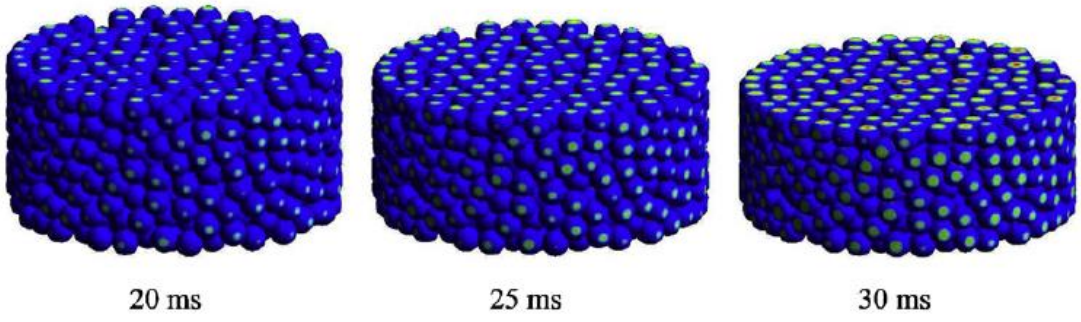

Figure 17: Deformed state of many discrete elements during compaction simulation proposed in [36]. Image taken from same source.

In many ways, the algorithm presented by Frenning makes sense to apply to the problem encountered in this thesis. Namely, potentially intersecting nodes are identified based on their proximity to nearby particles, then a restoring force is

applied to any nodes that are found to be in contact. This method, however, is still not entirely applicable to the contact model that this thesis describes. First, in using the distance between particles as the method for locating potential intersections, efficiency is low due to the computational expense of the square root operation necessary for calculating distance. Second, because the particles in this thesis are allowed to convect freely, the contact force calculation used in Frenning's work is not applicable. Frenning's simulations calculate the force applied to a node based on what force is required to keep the particles in contact because they are being compacted. Also, this force is directly related to the amount of penetration into the second body, or so-called "slave element". The problem in using this as a measure of force is that each node is not necessarily restricted to entering and exiting one element as would be the case in compaction simulations. This is further discussed in Section 4.3.1. However, the framework of Frenning's simulations is of particular interest to the application in this thesis. The code presented first attempts to selectively reduce the number of contact checks then applies forces based on the collision geometry. Thus, we see that the use of the discrete element method is currently at a point in which a red blood cell collision model can be developed.

Section 3.4: Spatial Binning

Due to the high computational costs associated with detecting collisions as will be detailed in Section 4.2.1, a method of reducing the number of particles and elements that are checked has been a large area of research for many years [38-41]. Spatial binning can be classified into two main categories: Static binning and dynamic

binning. The type of binning that is used is heavily dependent on the problem that is being addressed as each has its own set of advantages and disadvantages.

A static mesh attempts to define the entire area or volume for the problem before the simulation begins as demonstrated in Figure 18. The boxes that divide the volume are chosen to be of a size that is convenient in collision detection. For example, if a number of rigid spheres are placed in a volume and that volume is then divided into the bounding boxes for spatial binning, the boxes can be chosen to have an edge length equal to the diameter of the spheres. This allows for the simulation to only need to check particles that are contained within adjacent bounding boxes.

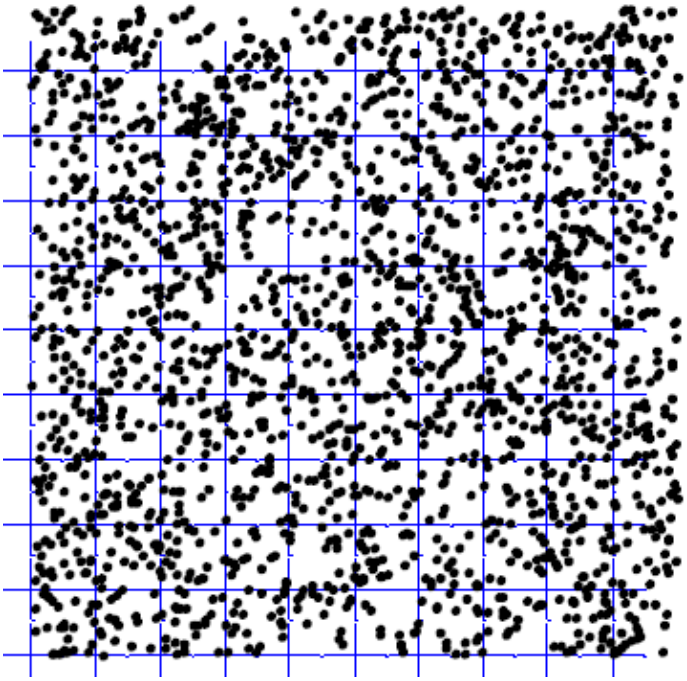

Figure 18: Example setup of static binning. Image Credit: Dr. Chris Chabalko.

While this method is highly effective in reducing the number of collision checks between particles, its main drawback is that setting up the bounding boxes require an initially large computation time.

Alternatively, a dynamic mesh allows for the creation of criteria based on the instantaneous location of the particle being searched. This type of search can take the form of only searching elements which are within a pre-prescribed distance in all three directions, so as to create a box that surrounds the node of interest. Another implementation is to reduce the number of nodes even considered for collision with elements by determining its location on the particle so that impossible collisions are ignored entirely. The dynamic mesh is useful in non-parallelized codes and is highly effective at reducing the number of checks for collision. The primary concern of a dynamic mesh is that as the scale of the simulation increases, its overall efficiency drops compared to the static mesh. However, when the simulation is relatively small, the dynamic mesh is both easier to implement, and more effective at reducing the number of computations needed. It is for this reason that the dynamic mesh will be used for the remainder of this thesis with the recommendation that a static mesh eventually be implemented as the scale of the simulations increase.

Section 3.5: Physics Considerations

Section 3.5.1: Conservation of Momentum

One of the most important considerations made during this derivation of force and direction for the collision modeling was that of the conservation of momentum [23]. That is, because the nodes on each blood cell are assumed to be of the same mass, the change in the total velocities of the nodes on one cell due to the collision must be equal but opposite in direction of the other cell in order to satisfy the conservation of momentum principle to ensure a physically accurate system.

$$\sum_{N_i} mv_i^0 + \sum_{N_j} mv_j^0 = \sum_{N_i} mv_i^f + \sum_{N_j} mv_j^f \tag{3.1}$$

Because all of the nodal masses are assumed equal, the masses can be cancelled from the above equation. If we examine the above equation with respect to the collision forces only, then the above equation becomes:

$$\sum_{N_i} \left(v_i^0 - v_i^f\right) + \sum_{N_j} \left(v_j^0 - v_j^f\right) = 0 \tag{3.2}$$

With the simplification of only examining the velocities that are due to the collisions, the velocity terms can be simplifed to $\Delta v = \frac{1}{2} a \Delta t$, then the above equation can simplify to:

$$\frac{1}{2} \sum_{N_i} a_i \Delta t = -\frac{1}{2} \sum_{N_j} a_j \Delta t \tag{3.3}$$

Finally, the simplification of only looking at collision forces leads to the substitution of $a = \frac{F}{m}$. Cancelling out like terms gives us a physically meaningful equation:

$$\sum_{N_i} F_i = -\sum_{N_j} F_j \tag{3.4}$$

Thus, in order to satisfy the conservation of momentum, the sum of forces applied to the first body must be opposite in magnitude to the forces applied to the second body.

Section 3.5.2: Normal Forces

In order to apply the conservation of momentum principles developed earlier, it made sense to attempt to develop a single total collision force that could then be easily distributed evenly between the two blood cells then distributed further between all of

the intersecting nodes on that body. It was originally believed that depth of intersection would be a reasonable parameter to consider when determining the normal force as is the case in [37]. However, because the cell is highly deformable and the surface is not easily defined by an analytic surface, tracking penetration depth becomes exceedingly difficult. While determining the distance from one intersecting node to the element it intersected seems trivial, there are actually several problems one encounters when trying to implement this method. The first problem is that it then becomes necessary to track which element the intersecting node crossed through which necessitates the addition of another data structure. While the data structure is simple and is easy to track and update, it adds unnecessary data that must be kept throughout the simulations. The primary concern with this extra data structure will be explained in Section 4.3.1, but adding an additional data structure, however simplistic, may lead to large jumps in computational time as the simulation scale increases. More concerning than this issue is that of the assumption made with such a method. If only the distance to the element that was originally intersected is calculated, then the possibility of a node exiting another element is discounted, particularly in cases where the red blood cells slide past each other. It can be seen in Section 5.2 that in angled collisions, it is actually very possible that a node exits another element which would change the penetration depth calculation. The only way to appropriately determine the penetration depth would be to check the intersected node against every element of the intersected body. Clearly this would cause a significant drop is computational efficiency, especially due to the finite nature of the elemental planes. It is extremely undesirable to compute the distance to the

center of each element because of the necessity for squaring values and taking a square root, both of which are expensive computationally. Thus, a different method of determining the collision of force was developed.

The new idea for total collision force is based on the penalty method as discussed in [31]. The penalty method is based on the idea that the area between two intersecting bodies in a two-dimensional simulation (see Figure 19) is directly related to the deformation area and is therefore directly related to the force experienced by the intersecting nodes.

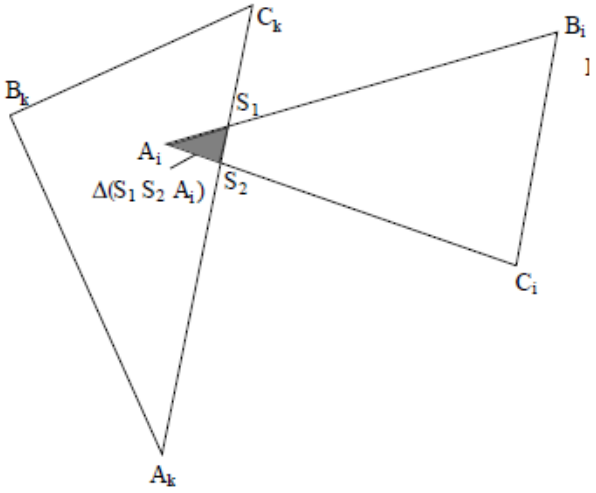

Figure 19: Overlap area used to determine contact force in 2-D simulation. Image from [27].

A common practice for the penalty method is to apply a damping force to the intersecting bodies based on their relative velocity, but because the calculation of relative velocity between intersecting nodes encounters the same computational problem as the penetration depth hypothesis, its implementation was avoided. In most simulations, this would lead to a conservation of energy and elastic collisions, but because of the membrane damping discussed earlier, the collisions will not be

elastic, which will be shown in Chapter 5. In expanding the penalty method to three dimensions, the assumption can now be made that the force exerted on a body is directly related to the volume of its intersection with another cell. For this thesis, the normal force applied to intersecting cells will be based on a constant pre-factor, which is varied to determine its effects in Chapter 5, and the intersection volume. The calculation of this intersection volume is developed and tested in Section 4.3.2.

Section 3.5.3: Direction of the Force

Upon developing the Finite-Discrete Element Method as it is seen in this dissertation, there are two major components that need to be derived in order to apply forces in a way that makes physical sense. To begin, we will review the two proposed directions that were explored and discuss the physics of each method.

The first direction that was proposed dictated that the force being applied to the collided nodes is applied in the direction from the center of mass of the collision volume to the center of mass of the blood cell as in Figure 20.

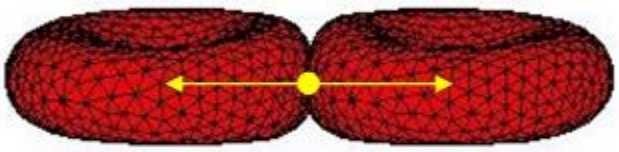

Figure 20: Proposed force directionality for two cells in simple contact. The yellow arrows point from the center of mass the collision to the center of mass of each cell. The conservation of momentum criterion appears to be satisfied in this case.

In a standard case where the blood cells approach each other directly, this method makes good physical sense in that the force is applied in exactly the opposite direction of the motion of the bodies. However, upon modifying the test case, the problem with this method becomes more apparent. In examining a case in which the cells travel in the same direction (see Figure 21) and begin to contact each other, the direction of the force tends to point in the opposite direction of the velocity, but does very little to push the cells away from each other.

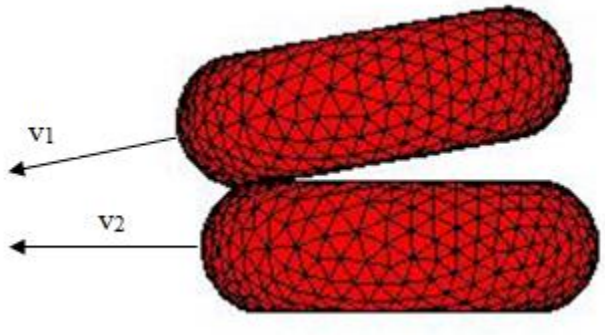

Figure 21: Test case for determining appropriate normal force directionality.

This creates a situation in which the cells slow down and "stick" to one another, which is not what is typically expected in this type of collision, though it still has some physical merit as this force could be reasoned to resemble frictional force during sliding contact. Still, in an effort to create a more appropriate physical system for collisions, a new method for determining directionality was proposed. In the first method, it can also be clearly seen that the resultant forces do not satisfy a force balance at the collision. In fact, the resultant force gives a net force (see Figure 22) to the right which clearly violates the conservation of momentum.

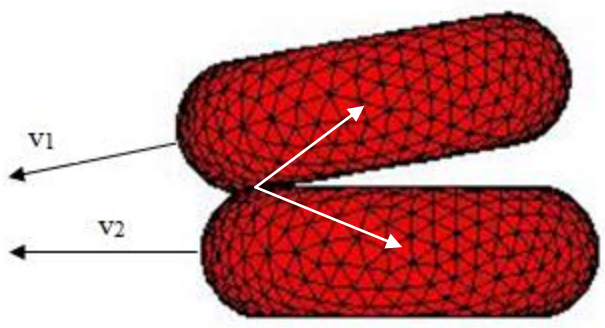

Figure 22: Resultant force vectors for the center of mass direction method. It is clear that there is a net rightward force.

To ensure that the second method satisfies the conservation of momentum, we will examine a similar scenario.

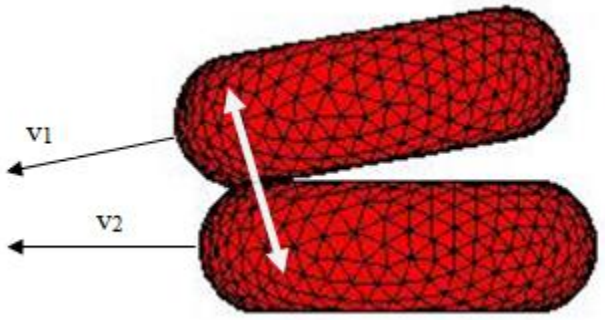

Figure 23: Angled collision test case for determining appropriate normal force functionality. Arrows represent the proposed direction of the normal force.

The second method that was proposed was to apply forces based on the locations of the center of masses of the entire cells involved in the collision as shown in Figure 23. In a sense, the force being applied is acting as a repelling force between the two bodies such that the bodies are always being pushed apart. Physically, this more closely resembles a normal force that would be seen in a typical collision. In examining the two cases that were described above, we see more physically accurate predictions. With two blood cells approaching each other directly, the force is applied in the same direction as with the first method, which makes physical sense.

We must now look at the second case which caused somewhat inaccurate results for the first method. In this case, the resultant force serves solely to repel the two bodies from each other, which is much more physically accurate than the first method as we would expect much stronger normal force than the first method would allow.

With the second method, the force vectors, by definition, point in exactly opposite directions. It was defined earlier that the total force due to collision is equal for each colliding body. Therefore, we have satisfied the condition that the forces, overall, are equal and opposite. It should also be mentioned that the total force is distributed among the collided nodes for each body. It was originally believed that a single force value could be applied to every node in a collision with the direction changing depending on which cell the node belonged to. This does not, however, satisfy the conservation of momentum because it is very unlikely that cells would have the same number of intersecting nodes on each body throughout a collision. This becomes especially true when colliding cells do not use the same coarseness of mesh.

It follows that by taking the total force described in the previous section and dividing by the total number of nodes from the cell that the force is being applied to, although the forces on each node for the different bodies will be different, the total force remains unchanged. Regardless of the number of points that comprise the surface, the net force will remain unchanged. At the point of writing this code, all blood cells have the same number of nodes, so adjustments for collisions between two cells with different numbers of nodes are not considered. If different levels of coarseness are to

be considered, then the conservation of momentum must be adapted because the masses can no longer be simply cancelled. However, because the simulations in Chapter 5 are only run with meshes of equivalent coarseness, this additional derivation does not need to be considered.

Section 3.5.4: Magnitude of the Force

In a 2-dimensional application of the finite-discrete element method, the force of repulsion is directly proportional to the area of the overlap [31], which for overlapping circles, is trivial to calculate using simple geometry. This idea is extrapolated to 3-dimensional approaches by making the collision force directly proportional to the volume of the overlap as determined earlier in this thesis. The finite-discrete element method also dictates that the collision force be directly related to the Young's modulus of the materials involved in collision. In the case of blood cells colliding, the contribution due to the Young's modulus is made significantly easier because the bodies will have the same Young's modulus, removing the need for differentiating between contact forces when one body is softer than another [27]. However, calculation of a Young's modulus for a red blood cell is nontrivial because of the interaction between the rigid spectrin network, deformable membrane, and the fluid within [42, 43, 44]. Thus, the primary variable that is unable to be determined is the coefficient that acts as the spring constant upon collision. Chapter 5 will present a range of spring constants that reproduce realistic results. Once the total force resulting from the collision is determined by multiplying the spring constant by the volume of intersection, the force can be divided amongst the intersecting nodes on

each body as described in Section 3.5.1 and given the direction as defined in Section 3.5.3. With this, the collision force as presented in this thesis has been fully defined.

Section 3.5.5: Model Shortcomings

One of the primary problems that arise from this method is the difficulty in implementing a frictional force that is felt upon collision between two membranes. It is not uncommon to treat collisions as frictionless in certain applications, but an additional friction force is helpful in successfully recreating the dynamics of a collision in general, particularly in situations such as the one presented in Figure 24. It will become more apparent how this can affect the dynamics in simulations described in Section 5.2.

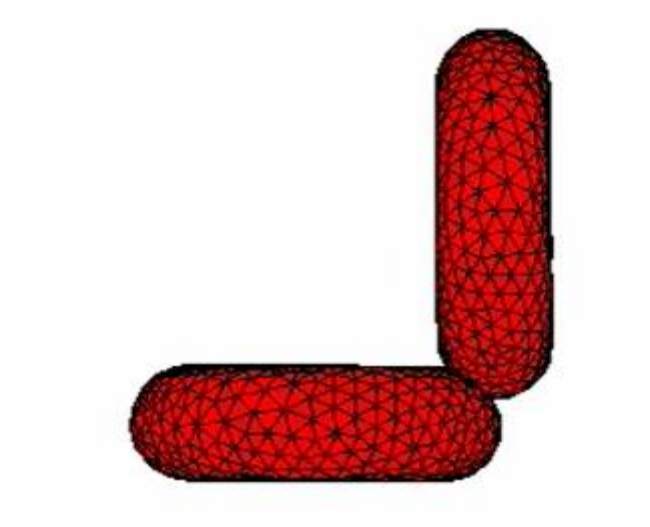

Figure 24: Capture of the rotated cell collision simulation presented in Section 5.2. Though the direction of the normal force to be applied is straightforward to calculate, the direction of the friction force is difficult to determine.

Based on the method of finding the direction of normal force, it is not straightforward how to apply a frictional force to colliding red blood cells. It is typically true that the frictional force is perpendicular to the normal force from a collision, but in three dimensions, this direction is nonspecific. The problem comes from the fact that there

is an infinite number of directions perpendicular to the normal force. While the direction is relatively simple to describe in examining the picture above, defining this direction in a simulation is not as simple. Further discussion on improving this model can be found in Section 5.4 and 7.

In addition to adding a frictional force, the model shows some weakness in conjunction with the damping model used. The damping model is written in such a way that large fluctuations in link length are minimized which is evident in Section 2.3. However, this link length fluctuation is what allows for the cell to rotate. The main effect that this damping force has on the dynamics in the simulations presented in Chapter 5 is that the damping effectively damps rotation out of the system completely in a relatively short amount of time. This can be seen in Section 5.2. It is clearly not realistic that two cells colliding in a vacuum as shown in that chapter should ever stop rotating after contact because there should be no force stopping the cells from doing so. The collision method gives a realistic rotation effect upon collision, but its overall effect on the dynamics of the system is diminished due to this particular implementation of the damping model. In order to more accurately recreate the dynamics of the cell in collisions that are not exactly direct as in Section 5.2, the damping model would need to be refined to allow for the transmission of the local deformation through the body which results in a rotational velocity that does not return to zero.

Chapter 4: FDEM Code Description

Section 4.1: Existing Code Description

Section 4.1.1: Code Architecture

The code that was inherited for modification was written and outlined by Ezzeldin [6]. The code was written with the flexibility to take several types of input and analyze strains on the surface of the blood cell. Ezzeldin then developed a theory for accumulated blood cell damage based on these strains. The code was written to effectively handle single blood cells surrounded by fluid in a closed vessel.

The code was written in Fortran90 for its efficiency in computationally-expensive applications. The main architecture of the code consists of a top-level directory that contains the MakeFile with two subdirectories that contain the source code, "SRC", and the simulation files and results, "Simulation". The code is run out of the file MAIN.F which contains the calls to initialize and begin running the simulations based on the flags supplied while compiling the code. All tasks that are completed by the code are written in subroutines that are also called from the MAIN.F file. This type of architecture makes adding entire subroutines to the code body relatively simple. Placing the Fortran files that contain the new subroutines in the running directory with the rest of the source code removes the need for interfacing the body of the code with the new subroutines because Fortran searches locally for subroutines when they are called in a program.

The code from Ezzeldin was able to compile through the use of a simple MakeFile that contains all of the flags and files that are necessary for running the code. Such flags would be those that link the code to established Fortran libraries that contain subroutines that are of use in certain subroutines. In particular, this code makes use of the LAPack, pppack, and BLAS libraries that are available for download freely on the internet. The MakeFile also contains the initial calls to all of the files that need to be linked from the SRC directory. From these links, objects are created as *.o files to be used from the *.exe file that is created. The executable file is placed in the Simulation directory where it can be run serially or as a parallelized code on a computing cluster.

There is a file contained in the Simulation directory that has all of the user-defined options. It is in this input data file that time step, number of time steps and other various parameters can be defined. It is in this file that the nodes' coordinates are defined, as well as where the elements are fully defined. The type of simulation to be run is specified in this file and more details on these options can be found in [6, 11]. Of particular importance in this file are the definitions of the spring types that are to be used in the simulations as well as the viscous parameters for membrane damping. The spring types and viscous damping equations are discussed in Section 1.3.

Section 4.1.2: Collision Detection

Prior to the work described in this paper, the Fortran code was not written to be able to handle collisions. As such, an overhaul of the code was necessary. The code had primarily been used to visualize single cells with pre-described paths or external

forces, so only inter- and intra-molecular forces such as those described in Section 2.3 and [11] needed to be considered. There was a small section of code dedicated to locating cell proximity, but it was written purely for the purpose of including a soft potential and was never fully developed and tested.

The existing method attempted to restrict checks between points and elements so that checks were only conducted when there was a small distance between the point and one of the points of the element. From there, if the dot product between the element normal and the vector between the point and point of the element changes signs before and after a time step, then a collision is assumed to have occurred. This method is reasonable for small numbers of cells, but as the number of cells becomes exceedingly large, the computation time to calculate the distance between the point and element would take extensive amounts of time. The method also has the potential to induce some error because the element is a finite plane, so a collision could be detected with this method even if a point does not cross through this finite plane. A new collision detection algorithm needs to be developed to improve efficiency and reduce the potential sources of error.

Section 4.1.3: Time Step Integration

The time step integration used in these simulations follows a modified verlet scheme in which the velocities are updated after the particle is allowed to convect. Symbolically, the particle is first allowed to convect:

$$r_i(t + \Delta t) = r_i(t) + v_i(t)\Delta t + \frac{1}{2} f_i(t)\Delta t^2 \tag{4.1}$$

Once the particle has been convected, the velocity is then found at an intermediate timestep:

$$\tilde{v}_i(t + \lambda\Delta t) = v_i(t) + \lambda\Delta t\, f_i(t) \tag{4.2}$$

Where λ is a value between zero and one that typically takes on the value of 0.5 to evaluate the new velocities at half of the time step. Once the new value of velocity at the partial time step has been calculated, this velocity can be used to calculate the forces that occur at the end of the time step more accurately:

$$f_i(t + \Delta t) = f(v_i(r(t + \Delta t), \tilde{v}_i(t + \Delta t))) \tag{4.3}$$

Now, the new forces can be used to give a more appropriate velocity at the end of the time step given as:

$$v_i(t + \Delta t) = v_i(t) + \frac{1}{2}\Delta t(f_i(t) + f_i(t + \Delta t)) \tag{4.4}$$

This new updated velocity can then be used to update the new positions as shown in equation 4.1.

This integration scheme will lead to some problems in implementing certain methods of collision modeling, which is discussed in Section 1.3.

Section 4.2: Collision Detection

Section 4.2.1: Node-Element Collision Detection

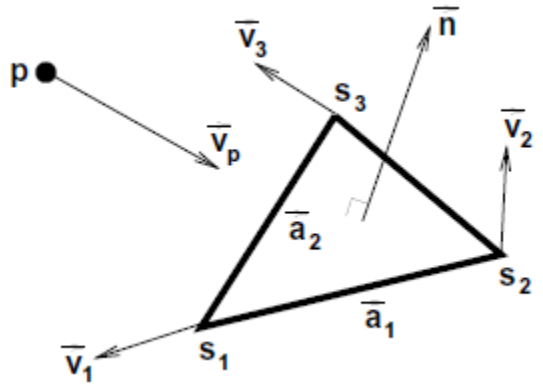

Figure 25: Physical representation of the node-element interaction to be checked for collision. Image from [13].

It is generally trivial to check whether or not a moving point crosses a plane defined by three moving points as shown in Figure 25. One needs to simply check if the sign of the dot product between the vector that points from the point of interest to the plane and the normal vector that defines the plane change after translation. The normal for each element is defined at any instant as the cross between the two side vectors, a_1 and a_2. Note that the side vectors are always defined such that $a_1 \times a_2$ results in a vector that points out of the body. The necessity for this consistency will become important later when the algorithm for tracking whether a particle enters or exits another cell is described. In order for this method to be correct, we must make the assumption that the velocity is constant throughout a time step. While this is not wholly true in the context of the integration method used in the simulations, an effective velocity is defined which satisfies the assumption. The concept of this effective velocity will be covered in Section 4.4.3. Symbolically, to determine if the

point has crossed the infinite plane formed by the element being examined, we use the following expressions:

$$b_0 = n_0 \cdot (p_0 - s_{01}) \tag{4.5}$$

$$b_{\Delta t} = n_{\Delta t} \cdot (p_{\Delta t} - s_{1\Delta t}) \tag{4.6}$$

Where a *0* subscript denotes the initial parameters that define the geometry, while a Δt subscript indicates position after the particles convect. If the product of these two quantities is greater than zero, then the point has remained on the same side of the element's plane which would indicate that no collision has occurred. If $b^* b_{\Delta t}$ is less than zero, then a collision between the point and the infinite plane formed by the element has occurred.

This check is clearly not sufficient to determine if a point has crossed the finite plane that comprises the element because the element has only been treated as an infinite plane to this point. Therefore, it is necessary to determine if this intersection occurs within the finite triangle defined by the points of the element. The algorithm for this routine was adapted from Fedosov [13]. This method involves reducing the geometric problem to a cubic equation so that the Newton-Raphson method can be used to find the exact time the point crosses the infinite plane. If we first define the equation of each edge vector as:

$$a_i = (s_{0i} + v_{0i}\Delta t) - (s_{01} + v_{01}\Delta t), \qquad i = 2,3 \tag{4.7}$$

$$a_i = a_{0i} + (v_{0i}\Delta t - v_{01}\Delta t), \qquad i = 2,3 \tag{4.8}$$

For brevity and consistency with Fedosov's nomenclature, the second term

$v_{0i} - v_{01}$ will be renamed to d_i. At any given moment then, the normal of the element is:

$$n = a_2 \times a_3 \tag{4.9}$$

If we insert the symbolic form of a_i into the above equation, then the resulting quadratic function is obtained:

$$n = a_{02} \times a_{03} + a_{02} \times d_3 \Delta t + d_2 \times a_{03} \Delta t + d_2 \times d_3 \Delta t^2 \tag{4.10}$$

The final step in creating the trinomial for use in the Newton-Raphson method is to relate this element geometry to the point of interest. This relationship is relatively trivial in finding a collision when one realizes that at the exact point of collision, the vector between the point of interest and one of the points is perpendicular to the element normal. By definition, to check for perpendicularity, the dot product between the two vectors must be zero, so:

$$n \cdot (p - s_1) = n \cdot (p_0 + v_p \Delta t - s_{01} - v_{01} \Delta t) = 0 \tag{4.11}$$

The last step in creating our polynomial is to group terms of equal magnitude of Δt together. In doing so, we obtain the following coefficients:

$$C_0 = n_0 \cdot (p_0 - s_{01}) \tag{4.12}$$

$$C_1 = \left[n_0 \cdot \left(v_p - v_{01}\right)\right] + [(a_{02} \times d_3) + (d_2 \times a_{03})] \cdot (p_0 - s_{01}) \tag{4.13}$$

$$C_2 = (d_2 \times d_3) \cdot (p_0 - s_1) + [(a_{02} \times d_3) + (d_3 \times a_{02})] \cdot (v_p - v_{01}) \tag{4.14}$$

$$C_3 = (d_2 \times d_3) \cdot (v_p - v_{01}) \tag{4.15}$$

Where the subscript of each *C* corresponds to the order of the Δt term. Now that a trinomial has been established, the integration time step can be divided into any number of separate time steps.

To use the Newton-Raphson method, each subdivision can be checked to find where the trinomial changes signs. That subdivision may then be divided into further subdivisions. This process can be performed any number of times until the right side of equation 4.11 is less than a predefined tolerance. The subdivision at which the right hand side of the equation is less than the predefined tolerance is then assumed to be the approximate time of collision between the point and element plane.

Once the time of collision has been found, parameters may be calculated to determine whether or not the collision occurs in the finite plane. This is accomplished by finding the projection of the vector at the approximate time of impact that is defined by the point of interest and point 1 onto the 2 edge vectors a_1 and a_2 as shown in Figure 26.

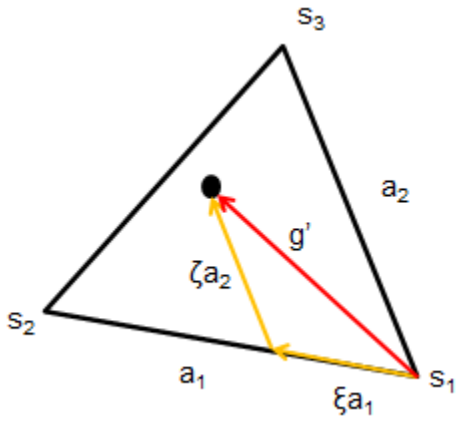

Figure 26: Projection vectors used to determine point-element intersection,

$$g' = p - s_1 \tag{4.16}$$

$$g' = \xi a_1 + \zeta a_2 \tag{4.17}$$

$$\xi = \frac{(g' \cdot a_1)|a_2|^2 - (g' \cdot a_2)(a_1 \cdot a_2)}{|a_1|^2|a_2|^2 - (a_1 \cdot a_2)^2} \tag{4.18}$$

$$\zeta = \frac{(g' \cdot a_2)|a_1|^2 - (g' \cdot a_1)(a_1 \cdot a_2)}{|a_1|^2|a_2|^2 - (a_1 \cdot a_2)^2} \tag{4.19}$$

If the two parameters, ξ and ζ, sum to be less than or equal to 1, then the point falls within the parallelepiped defined by the vectors a_1 and a_2 centered where the two vectors meet.

If each of the parameters, ζ and ξ, are also greater than zero, then the point passes through the finite plane and a collision has happened during the time step. For the purpose of this simulation, the exact time of collision will not need to be stored, so the detecting of the collision can stop after this check.

Section 4.2.2: Spatial Binning

Many of the methods proposed earlier make sense for large-scale simulations of red blood cells where the code is highly parallelized and communication between different processors can severely inhibit how quickly the code can run. In order to test the finite-discrete element method as it pertains to individual collisions, lesser implementations of binning can be applied to make the code run in a reasonable amount of time without having to significantly alter the pre-existing code structure. The code, with only the efficiency improvements described below, runs about 30 times slower than the code without collision detection in simulations with just two blood cells. To massively improve the performance of the code with a simple calculation, the collision method is only considered when the centers of mass of the

blood cells being examined are within a distance slightly larger than the maximum diameter of the red blood cells. This single computation can completely remove the collision detection from the simulation under certain circumstances, improving efficiency in the code by the factor of 30 that was mentioned previously.

Although the dot product takes a relatively short amount of computation time, it is still generally run as a subroutine. As such, it requires the passing of data between the main program and the subroutine. In an application in which even the interaction between just two cells can result in over a million checks, the dot product can significantly slow down the efficiency of the code. In order to reduce the number of calls to the dot product, and in order to reduce the total number of computations in general, it was desired to eliminate certain points from the list that should be checked as demonstrated in Figure 27.

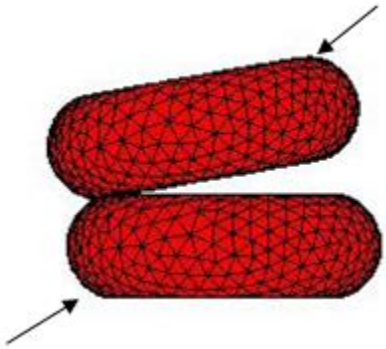

Figure 27: Representation of the necessity for computation reduction. Clearly nodes located at one arrow cannot intersect elements at the other arrow.

To accomplish this, it is possible to selectively reduce the particle list by comparing each point's location to the center of masses of both cells being checked for collision. As the model is defined, points that fall between the cells' centers of mass in any of the three directions are the only points that may possibly intersect the other cell. The

computation time associated with a greater than or less than boolean is significantly shorter than the calls to the dot product. The computation time was increased further if a point passes any infinite plane belonging to an element, regardless of how close the point and element actually were.

In order to reduce the number of elements that each appropriate point is checked against, a similar check could be conducted with the center of mass of each element. Those elements that fall between the centers of mass of the cells are passed into the next step of the simulation. However, a better method is proposed shortly hereafter. By only checking points that fall between the centers of mass, the number of checks can potentially be reduced by 50% (Figure 28), when the centers of mass are aligned nearly perfectly along two axes, effectively eliminating half of the points.

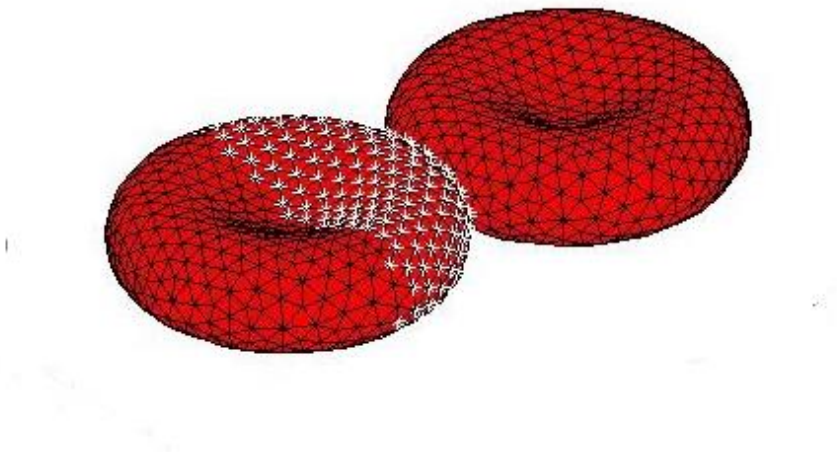

Figure 28: Figure showing the best case scenario of two cells colliding with the centers of mass perfectly aligned in two coordinates. In this case, only the half of the points that lay on the cell surface that falls between the centers of mass in the longitudinal direction should be checked for collision. These points are highlighted with white asterisks for clarity.

Even in the worst-case scenario, over 10% of points and elements are removed from the routine (Figure 29, Figure 30) which is significant considering the collision routine accounts for such a large portion of the overall code's computation time.

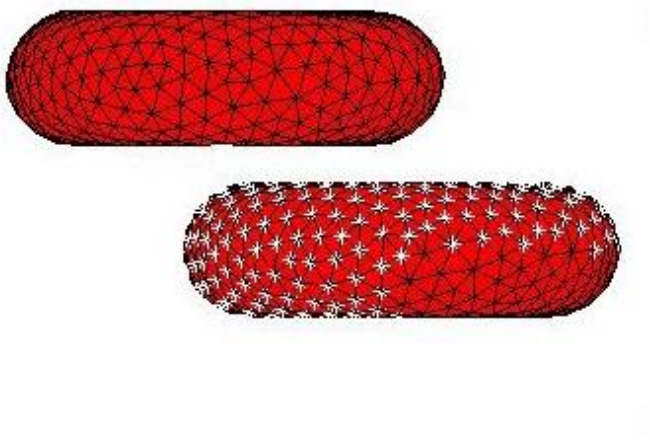

Figure 29: The worst case scenario that can be experienced when the centers of mass do not align in any way. The check, therefore, cannot eliminate a large number of points. The white asterisks indicate which points would be checked for collisions on the first cell against the elements of the second cell. Even in this worst case scenario, roughly an eighth of the points on the red blood cell can be removed from the check.

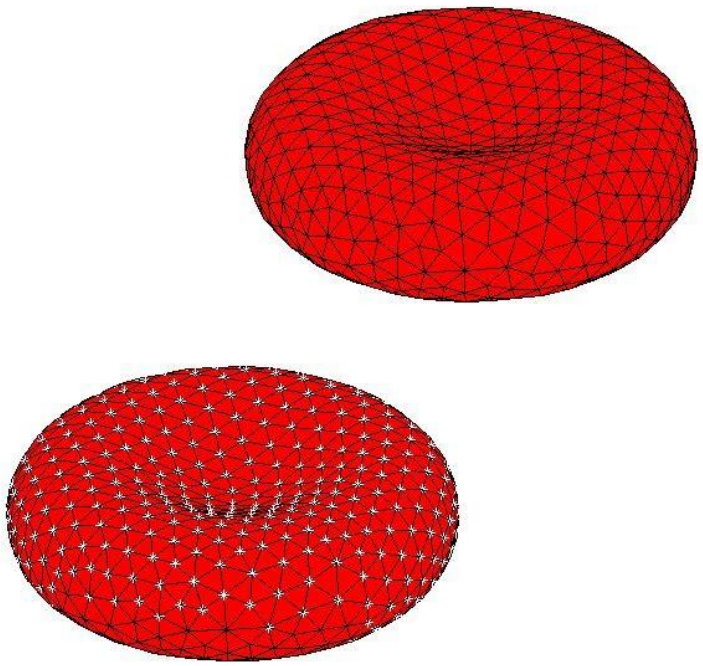

Figure 30: 3-dimensional view of the red blood cells as depicted in the previous figure. The white asterisks indicate which points would be checked against the elements of the second cell for collision. From this view, it can be seen that the only portion of the cell that does not get checked for collision is the bottom eighth of points.

To make the code even more efficient, a finer check is included that checks local proximity between the point and element (Figure 31). Each coordinate of each point is compared to the corresponding coordinate of the center of the element. If all three coordinates are within a prescribed distance that is relatively small, but still large enough to ensure that no collisions can be undetectable, then the collision check is performed.

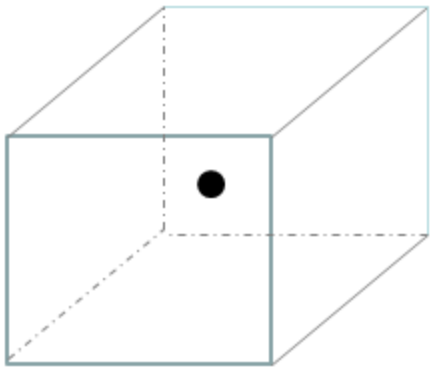

Figure 31: Local coordinate check for the dynamic mesh. Only elements whose center of mass falls within this bounding box are considered for collision. The bounding box is large enough that collision cannot possibly happen with an element that begins outside of this box.

This type of check is considered a dynamic binning method that works well in smaller-scale simulations such as the ones presented in this thesis. For large simulations with many blood cells, a static mesh is more desirable because it can effectively regulate the amount of data passed between processors when fully parallelized. A dynamic binning method is still useful in parallelized codes, but should ultimately happen after the cells have been run through a static mesh. It is the author's recommendation that when this code is run on a much larger scale, a static mesh precedes the center of mass and local proximity check by replacing the center of mass distance check.

A check similar to the one used for the points does not make sense for the elements because it is redundant with the local proximity search. In fact, the center of mass check takes more time computationally than the local proximity search and is less selective in its reduction, so using the center of mass check is extremely undesirable as the only method of computation reduction. It is worth noting that the local proximity search actually serves to eliminate all of the elements that the center of

mass check would remove from the elements with half as many computations. Though the addition and subtraction computations used are relatively trivial in small-scale codes, if the code is expanded to the scale of millions of cells, this improvement could prove significant.

At this point, it is helpful to describe why both checks for the point and element are performed instead of jumping directly to the local coordinate search. Visualizing the collision detection code as a flowchart as in Figure 32 below.

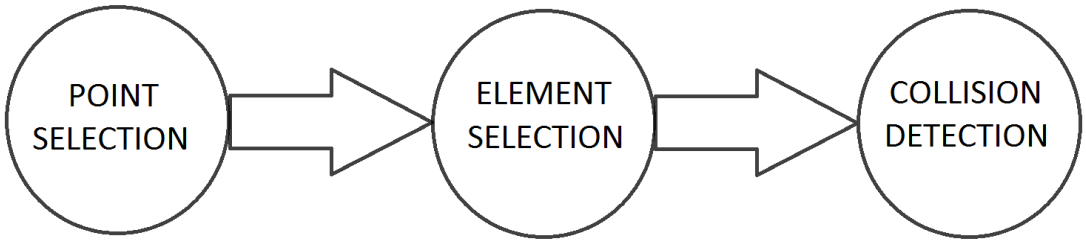

Figure 32: Basic flowchart of the collision detection method. The circles represent specific major routines in the code while the arrows represent areas between these routines that are convenient to break the code progression to prevent the code from carrying out an unnecessary check for collision detection.

In examining the code as a flowchart, the most computationally expensive portion of the code is the collision detection method itself, so it is advantageous to cause a break in the code when it does not make sense to check two points, such as when they are sufficiently far away. The break will stop the code from running through the rest of the flowchart. The earlier the code can be broken, the more advantageous it is to the efficiency of the code. As such, the initial point reduction, which is only three checks (one for each dimension), can reduce the total number of checks by the number of elements times the dimensionality of the system. In the case of the author's

simulations, this reduction was over 3,500 computations plus the computations regarding the actual collision detection.

The element check corresponds to breaking the second arrow of the flowchart which serves as the final check before the code begins the computationally expensive collision detection. While the second check does not reduce the number of computations as significantly as the first check, it still reduces the number of calls to the collision detection subroutine. This reduces the amount of data that must be passed between the main program and the subroutines, so it is still desirable to remove as many elements from consideration as possible. The local distance check successfully removes a large majority of elements from consideration without having to find the exact distance between point and element which is an extremely expensive computation. By including two checks between nearby blood cells, we have established that each check serves to significantly reduce the number of computations during the collision detection method by halting the execution of the collision detection at specific points during the routine.

Section 4.3: Principal Component Analysis Method for Determining Volume

Section 4.3.1: Problem Description

In dealing with continuous elements that are defined by discrete points, we are faced with the problem of how to approximate the continuous properties of the body based on the locations of these points. Indeed, there exist geometric arguments proposed by Fedosov [13] to calculate the volume of a closed, discrete body. The mathematics

involved is straightforward if the surface can be completely defined with triangular elements, as is the case with the red blood cells. Such volumes can be calculated using

$$V = \frac{1}{6}(\vec{\xi}^{k} \cdot \vec{t}_{c}^{k}) \tag{4.20}$$

Where $\vec{\xi}^{k}$ is the normal vector of the element of interest and $\vec{t}_{c}^{k}$ is the vector pointing from the center of mass of the entire body to the center of the element. This method is effective in determining the volume of a large body, such as the entire discretized red blood cell, where the entire body's connectivity is defined. In the case where the discrete volume is defined by two separate body's connectivity, there would be a significant challenge in establishing the connectivity of the points near the intersection junction. While local searching could make sense for developing a new connectivity, there would also need to be a sophisticated algorithm to prevent the new elements from overlapping fully defined elements that already existed. This type of algorithm is highly impractical, so alternative methods for volume calculation will be considered.

In looking at the potential size and shape of particle overlap, this method is not viable for use in the working code as it overlooks the partial elements that are part of the intersection. The use of $\vec{\xi}^{k}$ poses a problem with respect to code efficiency as it becomes necessary to track which elements have intersected, which involves searching the list of elements of a given red blood cell to see if every element is contained within another red blood cell. This can lead to an expensive algorithm that would necessarily attack the problem with a brute force method where only minor

shortcuts could be made which minimally affect efficiency. In addition, if an intersection is found, then the normal vector, center, and element index must all be passed, slowing the overall efficiency of the code.

Thus, we seek a method to estimate the total area, while maintaining efficiency for running the code. The following sections will explore proposed methods for dealing with this problem and comparing the accuracy of each method.

Section 4.3.2: Proposed Methods

The first proposed method will follow the approach that is described above. Though it is computationally expensive and generally inconvenient to program with regards to the amount of bookkeeping that is necessary, it serves as an excellent starting point for examining the potential methods that can be used to find volumes of shapes defined by arbitrary surface points. The primary weakness of this method pertains to the inability to measure elements that are only partially intersecting another body as seen in Figure 33.

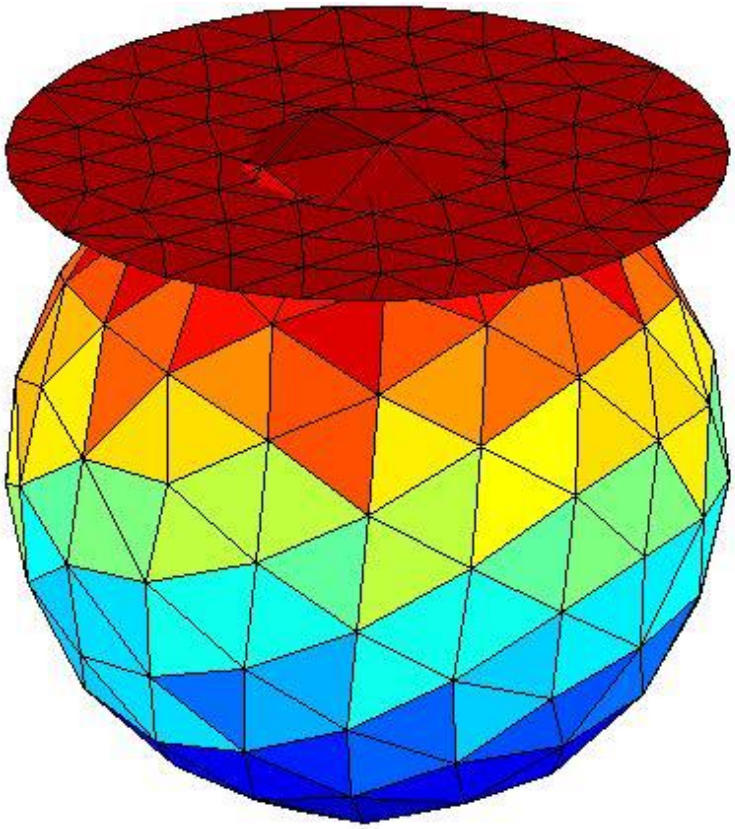

Figure 33: Example of partial element intersections in discretized collisions. Clearly, a large amount of the volume is defined by the partial elements.

Although partial elements would not be a legitimate concern when dealing with a body with many points and elements, collision contact often only involves a very small portion of the red blood cell's elements, so omitting partial elements can result in omitting a large portion of the intersection. This is particularly important in the beginning stages of red blood cell contact. If only full elements are calculated, there will a considerable jump in collision volume immediately after the first full element intersects and for every full element afterwards. These jumps in force will lead to unrealistic dynamics where the force between two bodies is discontinuous and the volume calculated will always be a significant underestimation of the total volume. As mentioned in the previous section, the need to search through the element matrix

would result in an inefficient method of checking each point that has intersected against each element in the entire body.

The second proposed method is based on the idea that elements that contain points that have intersected can be subdivided based on the number of respective nodes that are currently contained within another body. In other words, the volume due to the intersecting point can be adjusted such that the volume of each element contributes a portion of its total volume based on whether one, two, or three of its nodes have intersected:

$$V_{weighted} = \frac{N}{18}\left(\vec{\xi}^{k} \cdot \vec{t}_{c}^{k}\right) \tag{4.21}$$

Where N=1, 2, or 3 depending on the number of the specific element's nodes that are involved in the intersection. It is important to note that $\vec{t}_{c}^{k}$ refers to the center of mass of the intersecting elements that contain intersecting nodes. The method can account for partial elements by potentially overestimating and underestimating the partial elements. The approximation can give good results for volumes that are smaller or coarser than what can be handled by the first method. This method, in accounting for partial elements that have intersected, is much less prone to large fluctuations in volume and alleviates the issue of large jumps in force that are experienced in the first method.

However, as was alluded to above, this method becomes more accurate only as the volume becomes larger. In addition to its relatively poor approximation, the partial pyramidal volume calculation may actually serve to make the main program run even

more slowly than the unmodified method that was first presented. While the first method could be made more efficient by progressively checking each column, i.e. omitting a check of the second and third node of an element if the first node is not intersecting, this adjusted method would necessitate checking every node of every element against the list of intersected nodes to determine N in equation 4.21. Given that the red blood cell simulations are intended to be expanded to account for several million cells, passing large amounts of data between the main routine and subroutines is extremely undesirable. Because the centers, norms, and element indices must be available for this volume calculation and the use of equation 4.20, all of these data structures would need to be passed to the volume calculation subroutine. Therefore, the second volume approximation method, while more effective in calculating the volume, is inefficient in such a way that using this method is not feasible on a large scale simulation.

The third method proposed will be referred to as the Principal Component Analysis Volume Approximation, or PCAVA. The method works by approximating a surface of discrete points as an ellipsoid. In order to accomplish this, Principal Component Analysis, PCA, can be used to determine the approximate directionality of the principal radii. PCA is capable of receiving any number of 3-dimensional points as an input and outputting a 3-by-3 matrix containing the eigenvectors of unit length that will be used to calculate the approximate volume.

The input into the PCA method can be any *N* by 3 matrix, where *N* is any integer. For the purpose of this method, *N* will also be restricted to values greater than three. The physical reason for including this in the code is described in Appendix 2. One of the most desirable aspects of taking this approach is the non-necessity to pass large amounts of information to the collision volume calculation subroutine. In fact, the only data that needs to be passed to the subroutine is the set of coordinates corresponding to the points that have intersected between two bodies of interest. In the case of this method, there is not even a need for bookkeeping with respect to the format of these data points. The order of the nodal coordinates is not even important in the calculations because the connectivity of the points does not factor into the calculation of principal components. The overview of PCA is located in Appendix 2 at the end of this thesis.

After obtaining the unit eigenvectors pertaining to the principal components, we are still tasked with being able to make sense of these vectors. Two methods were proposed to determine the radii of the ellipsoid that is being approximated. The first method that will be discussed will involve the use of the dot product between the principal component vector and the vector between the collision's center of mass and each individual point. Demonstrated symbolically, this can be written as:

$$R_i = \max\left[PC_i \cdot \left(P_j - COM\right)\right], \quad (i = 1,2,3)\ (j = 1 \ldots intersect\ count) \quad (4.22)$$

Where R_i corresponds to the three radii that define the ellipse. Once all three of the radii have been calculated, the volume of the ellipse can be calculated:

$$V = \frac{4}{3}\pi * R_1 * R_2 * R_3 \tag{4.23}$$

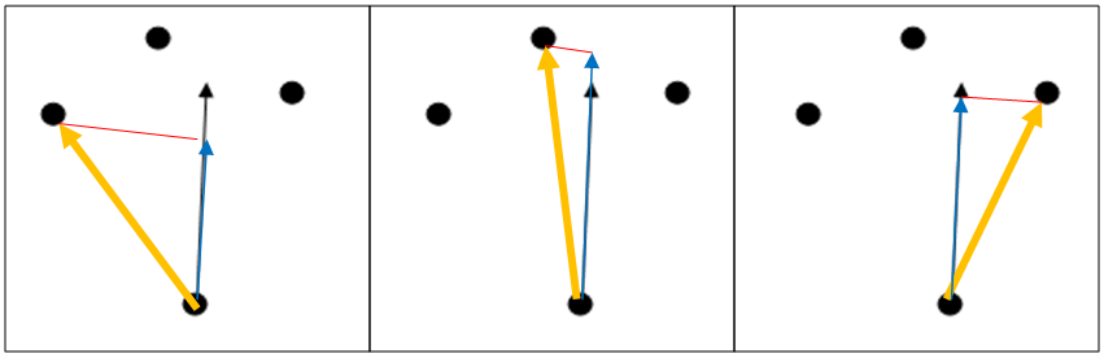

Figure 34: Dot Product method example. The component of each node vector (orange) in the eigenvector (black) direction is found (blue). The largest magnitude of the blue vector is taken to be the radius.

The dot product will be calculated for every point in the collision volume as shown in Figure 34. The point that results in the largest absolute value from the dot product would be stored. The distance from the center of mass to this point is then stored as the radius for the corresponding principal component vector. The dot product was used because it is a simplification of the component of the vector from a point to the center of mass onto the principal component vector:

$$comp_{PC_i}(P_j - COM) = \frac{PC_i \cdot (P_j - COM)}{|PC_i|} \tag{4.24}$$

Given that PC_i is a unit vector, the denominator is one, so the above equation simplifies to the dot product. In using the component, we are able to effectively weight each principal component based on the coordinates of the intersected nodes without having to worry about partial elements or irregular geometry. This is particularly important when there are a small number of points where certain principal components do not point towards nodes.

The fourth and final method that is being proposed for this work also involves the use of the principal components as defined in method three. The main change between the third and fourth methods is with regards to how the ellipse axes are selected. While the third method uses vector algebra to find which component of the point vectors with respect to the principal component is largest, the fourth method measures the angle between the principal components and the vectors between the collision's center of mass and each point.

$$\theta = \cos^{-1}\frac{a \cdot b}{|a||b|} \tag{4.25}$$

Calculating which vector results in the smallest angle is not trivial because the inverse cosine can result in values that range from 0 to just under π radians. Therefore, there must be an algorithm that is able to normalize each angle between 0 and $\frac{\pi}{2}$ radians. The angle must be normalized with respect to $\frac{\pi}{2}$ radians because an angle of π radians is perfectly aligned with the principal component, but in an opposite direction. These angles that are close to π radians should then be considered closer to alignment than those angles that may be smaller in magnitude, e.g. 179° should be considered closer than 3°. To account for the fact that the angles can be greater than $\frac{\pi}{2}$ radians, two separate checks needed to be included. The first check is used to find the angle if it falls between 0 and $\frac{\pi}{2}$ radians which follows equation 4.25 from above:

$$\theta_{ij} = \cos^{-1}\frac{PC_i \cdot (P_j - COM)}{\left|P_j - COM\right|} \tag{4.26}$$

where the magnitude of the principal component vector, $|PC_i|$, is omitted because it is of unit length. The second check, however, requires an additional step for normalization:

$$\theta_{ij} = \pi - \cos^{-1} \frac{PC_i \cdot (P_j - COM)}{|P_j - COM|} \tag{4.27}$$

Both θ_{ij}'s are checked against the θ_{ij}'s of all of the other points to select the smallest angles in each of the principal directions. The magnitude of the vector that corresponds to the smallest angle in each of the three principal component directions is then used as one of the radii for the ellipse.

Section 4.3.3: Testing and Results

In order to test the accuracy of each of the four methods, a simple test setup was devised and is shown in Figure 35 Figure 36. To be able to appropriately measure the actual area that should be measured, a known geometry was created and meshed with the use of distmesh [45] and made to intersect another known geometry that was also created and meshed through the use of distmesh.

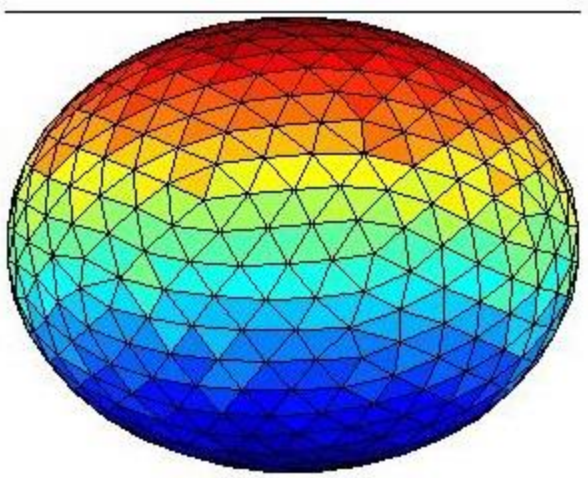

Figure 35: Initial test setup of the discretized sphere near a discretized wall. The discretized sphere is chosen so that the volume of the intersected spherical cap volume can be easily calculated.

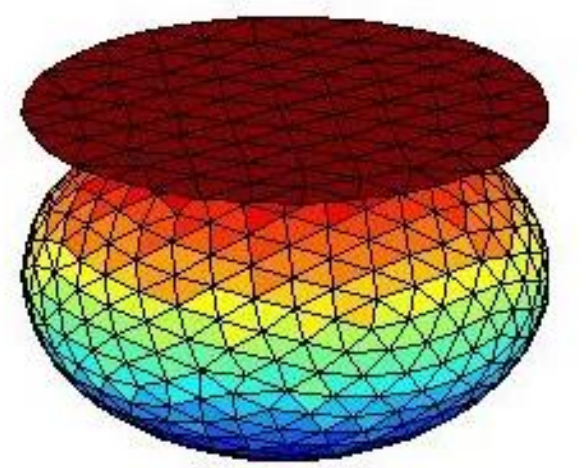

Figure 36: Three-dimensional view of the discretized sphere-wall collision test setup. In discretizing both shapes, we allow for the testing of all four proposed volume calculations in the same code for comparison against the theoretical volume of spherical caps.

In this simulation, a meshed sphere is allowed to intersect a meshed wall. Initially, the connectivity of the wall and sphere are generated in distmesh. This allows for the generation of the normal vectors of each element that are used both in collision

detection and volume calculation. This simulation does not deal in applying forces to allow for focusing solely on the volume of the intersection. The sphere is convected slowly in a direction perpendicular to the wall such that the volume of all four methods can be calculated consecutively in the same simulation. Because we are using a sphere with a flat plane intersecting, the volume of a spherical cap (Figure 37) can be used to calculate the theoretical value.

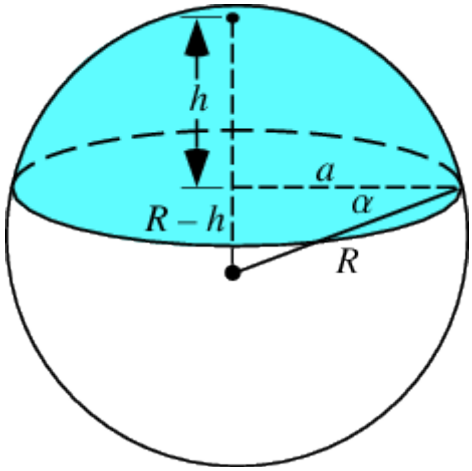

Figure 37: Picture showing a spherical cap from a sphere intersected by a plane. This geometric argument can be used to calculate the volume of intersection to compare against the volume approximations. Image from [46].

$$V_{cap} = \pi * \frac{h}{6}(3a^2 + h^2) \tag{4.28}$$

Because the radius of the sphere is constant, the value of h can be found simply by comparing the location of the center of mass of the sphere and the wall height. From the height and radius, the cap radius can be found through the Pythagorean theorem:

$$a = \sqrt{r^2 - (r - h)^2} \tag{4.29}$$

Specifically in this simulation, the radius of the sphere was chosen to be of unit length. Its surface was divided into 480 nodes by distmesh with 956 elements. The wall, similarly, was given a radius large enough to not be completely enclosed by the

sphere and divided into 88 nodes creating 143 elements. The following figure displays the data that was calculated for each volume calculation method.

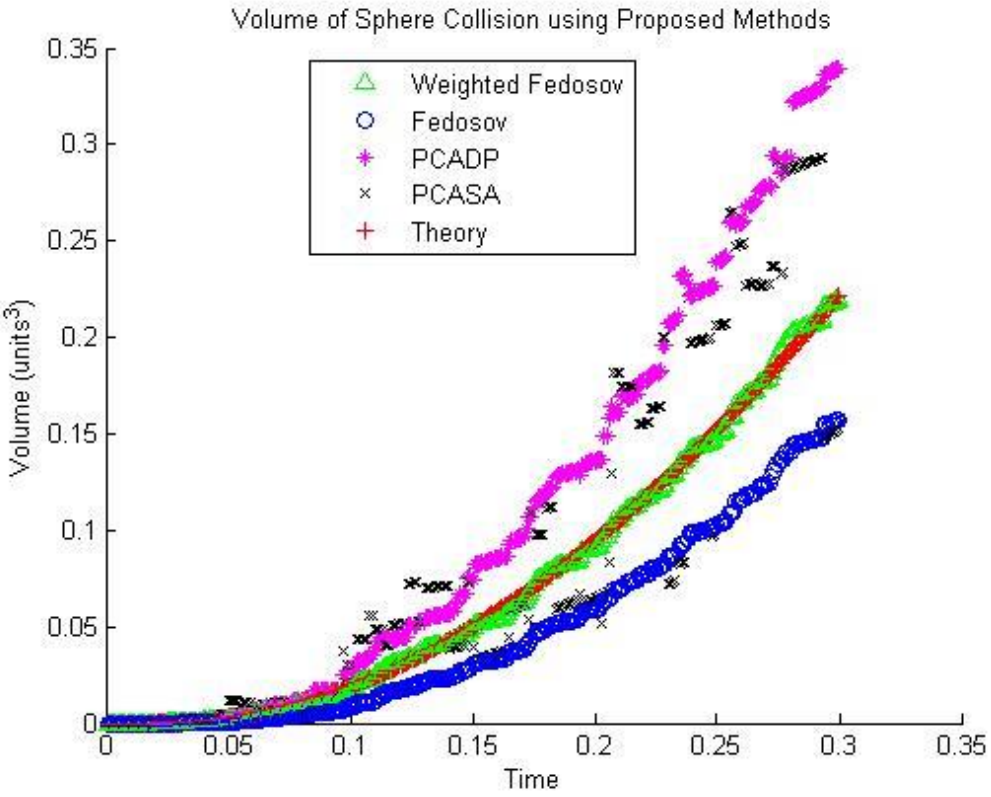

Figure 38: Calculated volumes for the four proposed methods for volume approximation compared against the theoretical values for volume.

It would appear from the results above that the Fedosov Method [13] consistently underestimates the volume of the collision. In addition, the PCA methods both overestimate the volume that is calculated. It is worth noting that the dot product method is significantly smoother than the Smallest Angle Method. The Smallest Angle Method shows clear steps in which the volume transitions between overestimating and underestimating, showing a lack of consistency. In this simulation, it is clear that the Weighted Fedosov Method most closely follows the theoretical value for volume, and a reasonable statement would be to conclude that

the Weighted Fedosov Method is the best method for approximating the volume of collisions.

This conclusion, however, does not consider the true scope of the implementation used in the following simulations. The volume calculated above accounts for 5% of the total sphere volume, which is very large for the discrete element method. If, instead, we examine only the beginning of the simulation (Figure 39), a different conclusion can be drawn as to which volume approximation methods are appropriate for use in the simulations that will follow in subsequent sections.

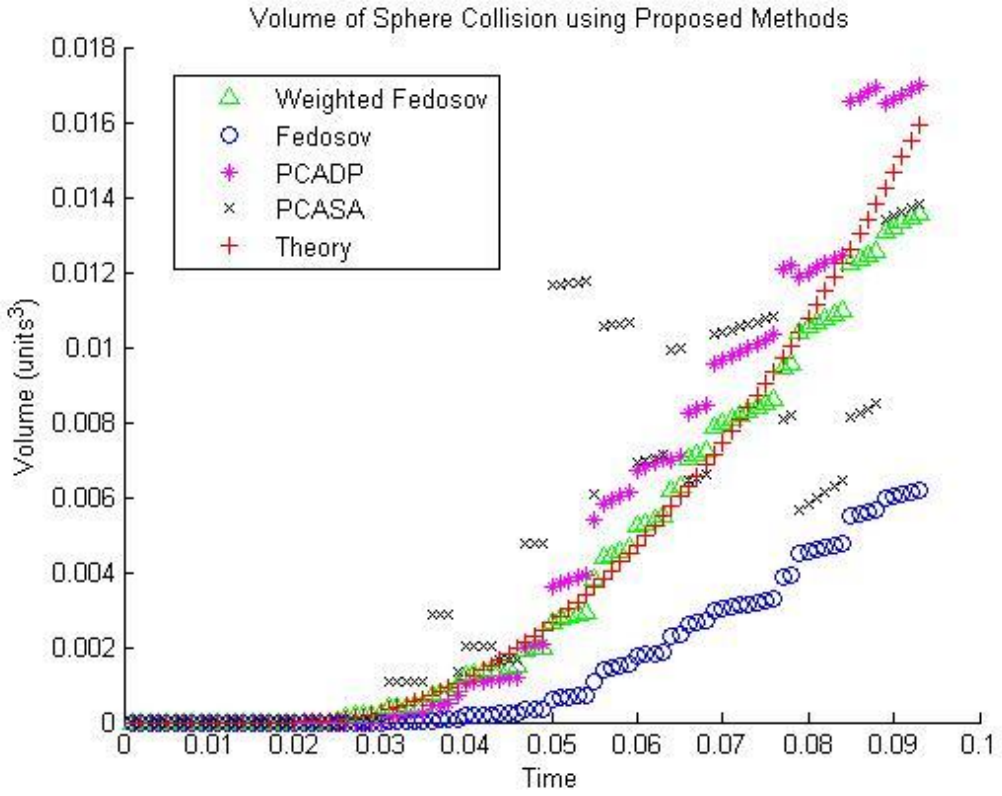

Figure 39: Smaller volume simulation showing calculated volumes for the four proposed methods for volume approximation compared against the theoretical values for volume.

In analyzing Figure 39, the conclusion that the Weighted Fedosov Method is the most appropriate method for volume approximation is no longer as clear. While it is clear

that the Weighted Fedosov Method is reasonably accurate in calculating arbitrary volumes, the PCA Dot Product Method also appears to be accurate for smaller volumes. The Fedosov Method and the PCA Smallest Angle Method still prove to be problematic in their implementation. The Fedosov Method continues to underestimate the areas consistently, while the PCA Smallest Angle Method is erratic in its predictions. For the remainder of this section, the Fedosov and PCA Smallest Angle Method will be disregarded.

Because both the Weighted Fedosov and PCA Dot Product Method work well in approximating discretized volumes, an argument must be made in choosing one or the other for the simulations. The argument has two major components to be made in favor of the PCA Dot Product Method. The first argument that will be made is with regard to the test run above. The area of the sphere is just more than 4 square units and has 480 nodes. The red blood cell has an area of approximately 135 square units with only 611 nodes on its surface. As the coarseness of the volume increases (Figure 40), the smoothness of both methods will be reduced, but the Weighted Fedosov Method is more susceptible to large jumps which are undesirable in trying to create accurate dynamics.

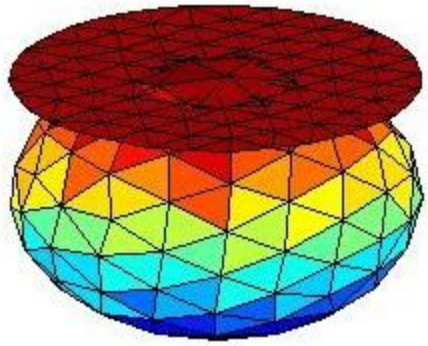

Figure 40: Coarser setup to test the final two methods against one another to see which is more appropriate for discrete element method simulations.

In changing the sphere's level of coarseness to about one quarter of its original mesh, we can check the validity of the two methods on a volume that more closely resembles the red blood cells used in later simulations. In this second test, the sphere now has only 126 nodes instead of the 480, while only having 248 elements. Though this mesh is still finer than the blood cell, it is the coarsest mesh that can be generated with distmesh before instability occurs. At the time of the writing of this thesis, distmesh was the only available method for creating a mesh. A coarser mesh reveals interesting trends (see Figure 41) that are not obvious from the finer mesh.

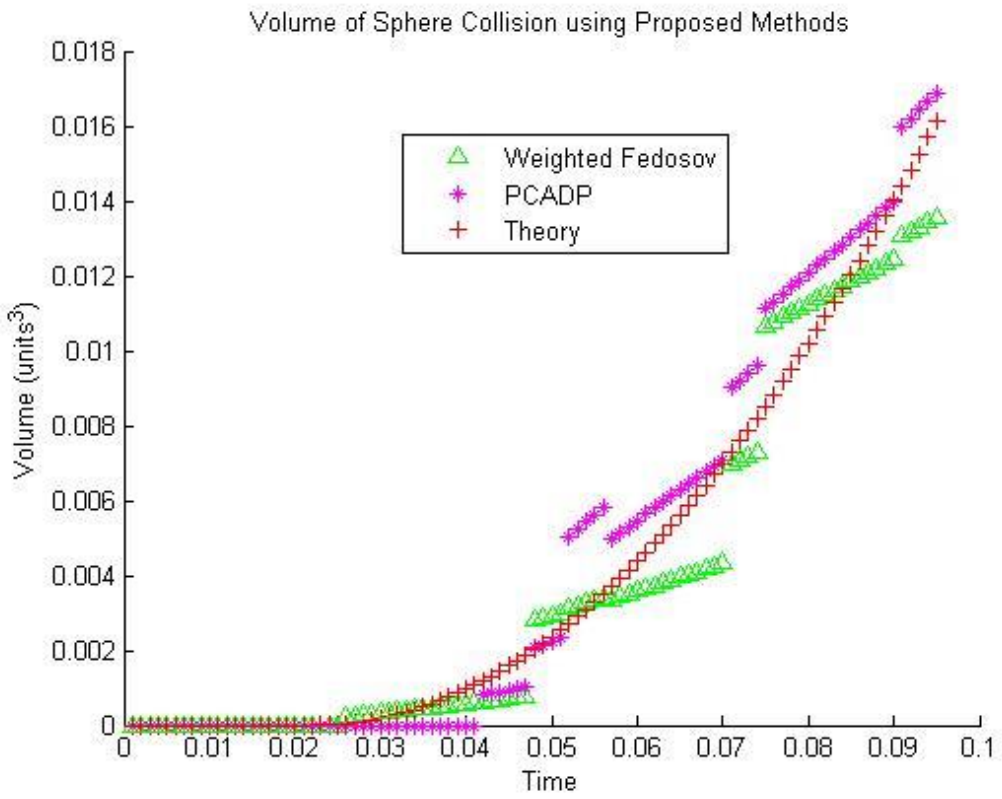

Figure 41: Volume approximation results using the Weighted Fedosov Method, the PCA Dot Product Method, and the theoretical value.

With the coarser mesh, we see large jumps in force from the Weighted Fedosov Method as was theorized earlier. While the PCA Dot Product Method also has the jumps in force, they are smaller as can be seen at time step t=.047. The large jumps that occur around t=.07 are what the simulations are trying to avoid to prevent large deformations instantaneously. Thus, as the mesh gets more coarse, the Weighted Fedosov Model becomes less practical from an accuracy standpoint. However, if the mesh becomes finer, the Weighted Fedosov model becomes more feasible and would be recommended in simulations where efficiency is not of great importance.

However, the second argument considers the long-term goals of these simulations as discussed in Section 2.2. The intention, as previously stated, of these simulations is

to eventually create and track millions of red blood cells as they flow freely through an artificial heart valve. Because of the extremely high computational cost of tracking the locations, velocities, and forces of each cell, the simulations are intended to be highly parallelized such that each cell can run on its own thread independently. In this scheme, passing data between threads slows down the threads of both cells involved. This is because the threads of the cells must stop computing while transferring data to a tertiary thread and waiting for the appropriate data to be returned. When the number of blood cells is on the scale of millions, passing even one extra matrix can result in significant slowing of the simulations. The volume calculation from the Weighted Fedosov method requires knowledge of the nodes and elements of both cells involved in collision detection. The PCA Dot Product Method only requires the elements of one cell which would allow one thread to only need to pass its nodal information. This distinction would potentially represent a significant improvement in efficiency.

In conclusion, both for the purposes of this simulation and for the eventual long-term goals of this research project, the PCA Dot Product Method is recommended for the measuring of volumes of colliding discretized bodies over the three other methods presented in this section.

Section 4.4: New Code Architecture

Section 4.4.1: Overview

Upon beginning this project, the Fortran code was written to appropriately handle several blood cells independently, but the code was not tested in this context. Some changes were made in order to allow for the correct handling of the blood cells. These changes were relatively minor considering the size of the code as a whole. Additionally, there was no way of handling cell-cell interactions. Minor modifications had to be made to the overall code structure to improve the ability to handle multiple cells. The following sections are provided in order to describe the architecture of the contact model and how it was implemented into the preexisting code.

Section 4.4.2: Architecture

The red blood cell code was written in such a way that there were global modules that were selectively used in each module and subroutine to reduce the amount of data that needed to be passed between *MAIN.F* and every other routine. In order to keep the code implementation consistent with the preexisting code, a module was created to contain all of the data that stores information about the collisions between cells. The inclusion of such a module allows for the existence global data that is only seen in the collision model, improving ease of use and efficiency. All subroutines and functions are stored independently to allow for easier debugging. Although this can make interfacing the program slightly more difficult, the benefits to debugging the body of the code greatly outweigh the necessity for interfaces.

In working with Fortran, there is a need to carry a larger number of variables than many other widely used languages due to its relatively rigid data structures. The allocation and deallocation processes necessary in Fortran for dynamic array sizes are extremely computationally expensive and should be avoided whenever possible for the sake of efficiency. Because dynamic allocation is being largely avoided, the top level program stores much of the data regarding collisions to be passed to lower programs in a way that would typically be redundant in other languages.

In total, there are 5 global arrays that store contact data. The array `intersectedNodes` acts as the primary storage array in which the indices of any nodes that have crossed another cell's surface are stored. It is a 3-dimensional matrix where the first index corresponds to the index of the intersecting cell, while the second index stores the index of the cell that is being intersected. The third index stores all of the indices of the nodes from the first index cell that have intersected the cell of the second index. Ordinarily, the third index would be dynamically allocated such that its dimension would be the same as the total number of intersected nodes so that superfluous memory usage would be avoided. However, as mentioned earlier, the dynamic allocation would take significantly longer than any benefits from reducing the array size, especially as the number of cells increases.

In order to then keep track of the number of nodes that have intersected between two bodies, another matrix is stored. The `intersect_count` array needs only be a 2-dimensional matrix where the first index corresponds to the intersecting cell and the

second index corresponds to the intersected cell, as was the case with the first matrix that stores nodal indices. However, unlike the first matrix, only the number of intersected nodes is stored at the respective index. This value is always updated immediately as the `intersectedNodes` array is updated such that the length of the third dimension in `intersectedNodes` matches the value of the corresponding index in `intersect_count`.

The third array, `intersect_vol`, is used in a manner where the indices are consistent with how they are used in `intersect_count`. In this 2-dimensional array, the components of the array are the calculated volumes resulting from the PCAVA method. These volumes are used to populate the fourth array, `intersectForce`. The components of this array are simply the components of `intersect_vol` multiplied by an effective spring constant, K_s, which is explained in section 3.5.4. The values in this array are scalars that correspond to what will be the magnitude of the total collision force on each of the bodies.

As was discussed in the section on conservation of momentum, an array, `FCollArray`, can be made that applies force to all of the nodes involved in a given intersection that satisfies the conservation of momentum. To be able to simply add this collision force with the rest of the forces experienced throughout the cell described in Section 1.2, the total collision force is divided between the intersecting nodes and stored in the fifth and final arrays. It is at this point that it is important to make the distinction that the first four arrays are global arrays that contain information as it relates to the entire

system. The final array that carries the nodal forces due to collision are not global in that each of the `FCollArray` arrays contains only information for one specific blood cell. Populating this array with the final resultant force from a collision allows the program to simply add the collision forces to the total force array that is a 2-dimensional array with dimensions (*Number of nodes*, 3), where the second index pertains to the dimensionality of the system. To populate the collision force array, the program defines the nodal force for each cell for each of its different collisions. The direction of the force is then defined as described in section 3.5.3. Lastly, the program stores the force in the nodal indices that are given in `intersectedNodes`.

A final step in this implementation must be considered where a cell may be intersecting multiple cells at the same node. In this case, the forces in `FCollArray` are actually the summation of all collision forces on a given cell. In most cases, the multiple collisions will be applied to separate nodes where this consideration becomes irrelevant, but there are special cases in multiple collisions where the consideration is non-trivial.

Section 4.4.3: Implementation

Implementing a collision model into a pre-existing red blood cell code presented a unique set of challenges. The challenges faced were primarily in determining the appropriate location to implement the collision model and in correctly formatting inputs and outputs to be usable in the pre-existing code modules. The following sections will detail the multiple changes and updates that had to be made in order to effectively complete the collision model.

The most difficult challenge faced in integrating the collision model came from the fact that the time step integration method was not straightforward. The details of the integration method can be found in section 4.1.3. The main point of the integration method is that the velocity stored by the program is not the velocity used in moving the red blood cells. This discrepancy becomes an issue in the collision detection method. Because the method depends on the true velocities of the particles to be able to track which particles pass through finite planes, there needed to be a way of storing an effective velocity for each node that was not dependent on either of the two velocities that are used in the current time step integration method.

In order to accomplish this effective velocity, the program must allow all of the particles to be advanced with the usual integration method. The current implementation of the integration method has two arrays that store the initial and final positions of all of the particles. Just before the end of the time step integration, the initial positions array is replaced by the final positions array, so a step can be taken before this final overwrite that finds the movement of the particles throughout a time step. An effective velocity array is created by taking the difference between initial and final positions.

$$v_{eff} = \frac{\left(s_f - s_0\right)}{\Delta t} \tag{4.30}$$

It is assumed that the velocity between the beginning and end of the time step is constant. This is a fair assumption because even though the velocities are updated at the end of every time step, the velocities throughout the time step are considered

constant. The velocities that are stored in the program, however, do not correspond to how the particle travels. There is an acceleration component due to the force that contributes to particle motion:

$$s_f = s_i + v_i \Delta t + \frac{1}{2}\left(\frac{F}{m}\right)\Delta t^2 \tag{4.31}$$

where m=1 for each particle in the unitless simulations.

Although creating an extra array is a detriment to the efficiency of the entire code, this was believed to be the only way to determine the actual motion of the particles without relying on parameters from the integration scheme. This effective velocity is passed as the velocity parameters of the respective nodes and elements.

There is particular difficulty faced when trying to integrate the collision force into the time step integration method because of its averaging effect from multiple time steps. Because the particles have to move before the effective velocity can be calculated, the calculations for detecting collisions have an implicit latency.

$$VEL1 = VEL + \lambda * \Delta t * (F_{Tot} + F_{CollArray}) \tag{4.32}$$

The $F_{CollArray}$ term contains the collision force from the previous time step because *VEL1* corresponds to the velocity at a fraction, λ, through the time step. It would be unreasonable to compute the collision method twice per time step because it already accounts for more than ten times the computation time compared to the code without the collision method. Because the collision method can only be realistically run once per time step, the collision force at a fraction of the way through the time step is

assumed to be approximately close to the collision force at the beginning of the time step.

The forces that result from collisions are felt only partially during the time step that the actual collision occurs. The partial effect arises from the time integration where velocity is updated at the end of the time step:

$$VEL2 = VEL + .5 * \Delta t * (F_{Tot} + F_{CollArray} + F_{Tot_{Old}}) \quad (4.33)$$

The $F_{CollArray}$ in the above equation corresponds to the collision force array that arises from the collision after the particles have convected. This latency is considered to only be a small effect because the time step is relatively small such that there is not a large jump in force upon initial contact at the second time step. Additionally, a small time step would necessitate that the area of the collision upon initial contact would be small. This would lead to a collision force that would also be small enough such that the dynamics of the total system would be relatively unaffected. The effect of the latency where the collision force is always applied a half time step too late is minimized by a small time step as well.

Chapter 5: Test Results and Analysis

Section 5.1: Direct Collision

In order to be able to check the effectiveness of this method, several simple simulations were proposed to be able to visualize the cells' dynamics during collisions. Two identical cells were generated using GAMBIT consisting of 611 nodes which generate 1218 elements. To test the collision method, one of the particles is kept stationary, while the other is given a total velocity in the direction toward the first cell as shown in Figure 42.

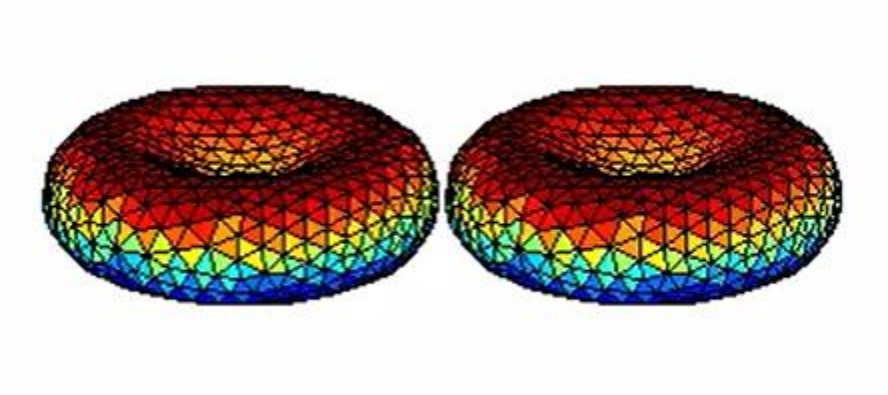

Figure 42: Initial test setup of the direct collision. The particles are exact clones of each other with an offset at the beginning of the simulations. The cell on the left is then given a velocity in the direction towards the right cell.

The initial mesh that was generated is close to the equilibrium state of the cell, but the energy is not fully minimized. To account for this, the cells are allowed to minimize briefly before the cell is given velocity. During the simulation, the particle positions, velocities, energies, and cell centers of masses are all recorded every 25 time steps. Recording more often would lead to an unnecessary amount of data and slow the efficiency of the code. If this data was to be recorded less often, there would be a risk of omitting important data that may occur between data recording.

For these simulations, the *K_s* parameter for defining the magnitude of the force was held constant through each simulation, then varied to see how the *K_s* parameter can affect the dynamics of a red blood cell collision using the Discrete Element Method. Note that the bending, area, and volume potential energy terms are omitted in the results section because the fluctuations in these terms are on a considerably smaller scale than the spring potential energy term. The spring potential energy term will be considered the most important term for determining stability in the system. Kinetic energy and total system energy will be displayed to demonstrate the conservation of energy that is important to satisfy in discrete element method simulations.

Section 5.1.1: Energy Conservation and Dissipation

The data presented below corresponds to *K_s* values of 10, 75, 100, 200, 300, and 500 respectively.

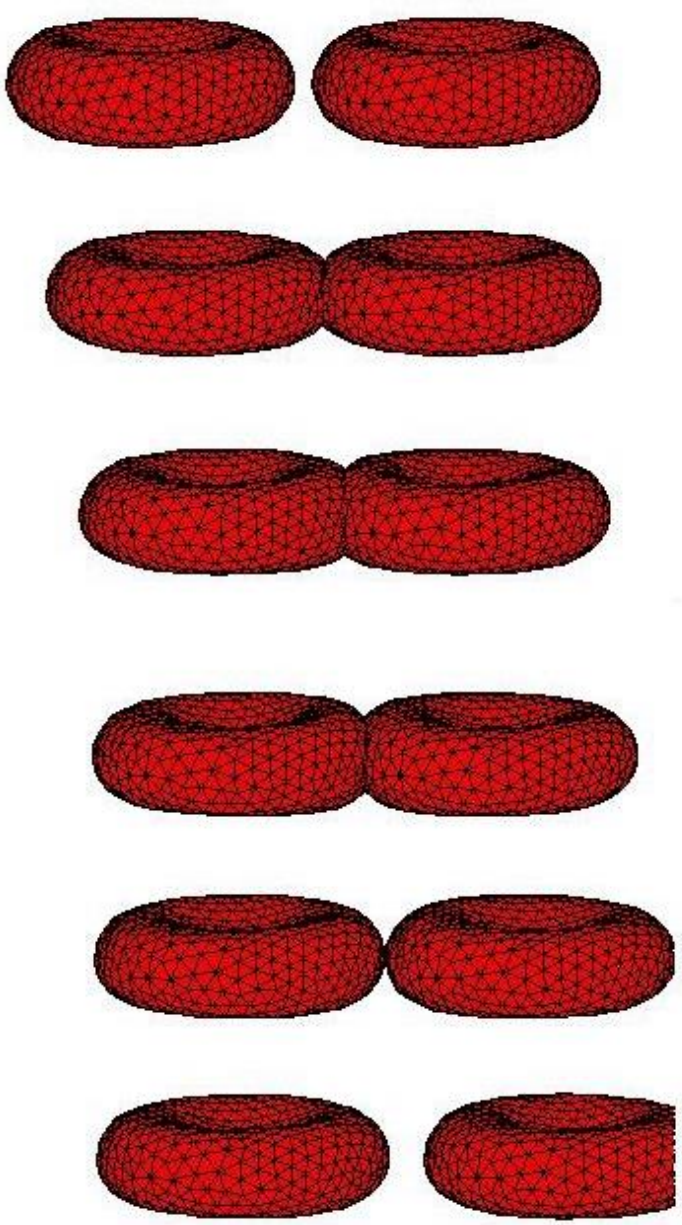

Figure 43: Collision progression of two red blood cells with *K_s*=10. Each progression represents Δt=28.4ms.

To analyze how the *K_s*=10 simulation fares, further data must be presented, but it makes sense to first discuss the empirical results from the above figure. One of the

most notable aspects of this simulation is that the cells intersect for a period of time before there is any visible interaction from the collision force. The collision is gradual and local deformation is minimal which is not what would typically be expected in collisions. This is discussed in detail later in this chapter.

The collision should still be analyzed with respect to its velocities and energy (Figure 44 - Figure 47) to determine if the simulation is physically feasible.

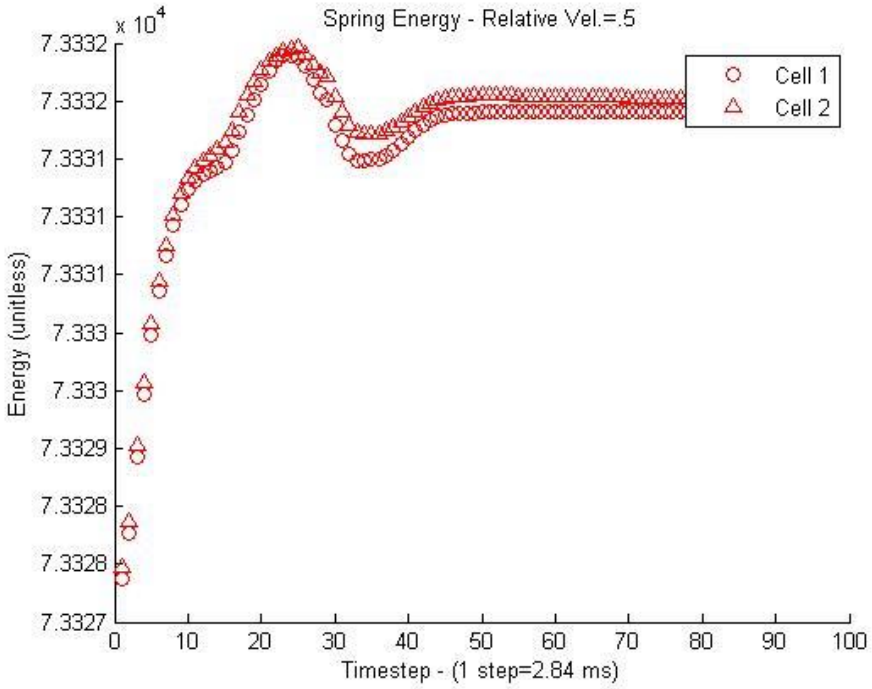

Figure 44: Spring energy of *K_s*=10 direct collision simulation.

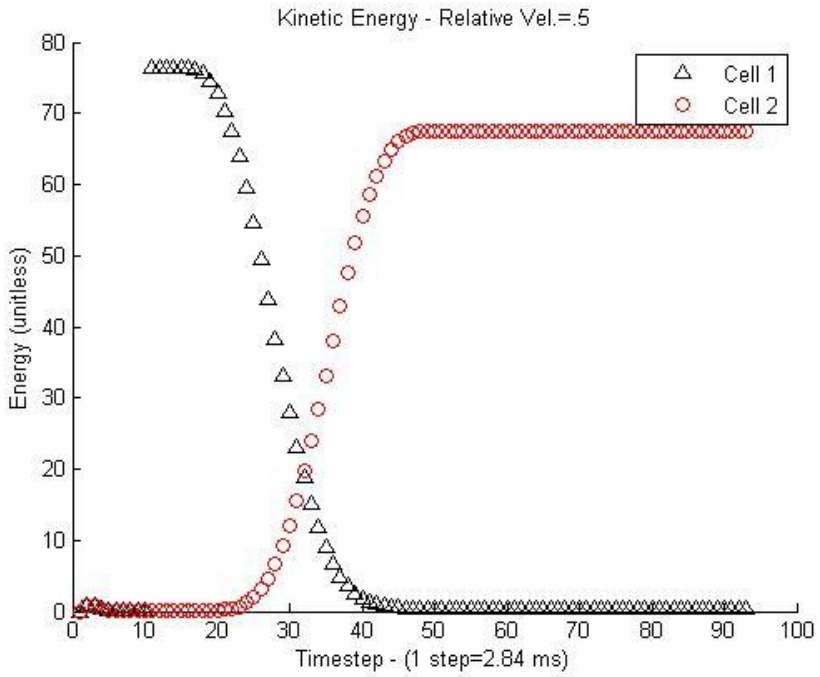

Figure 45: Kinetic energy of *K_s*=10 direct collision simulation.

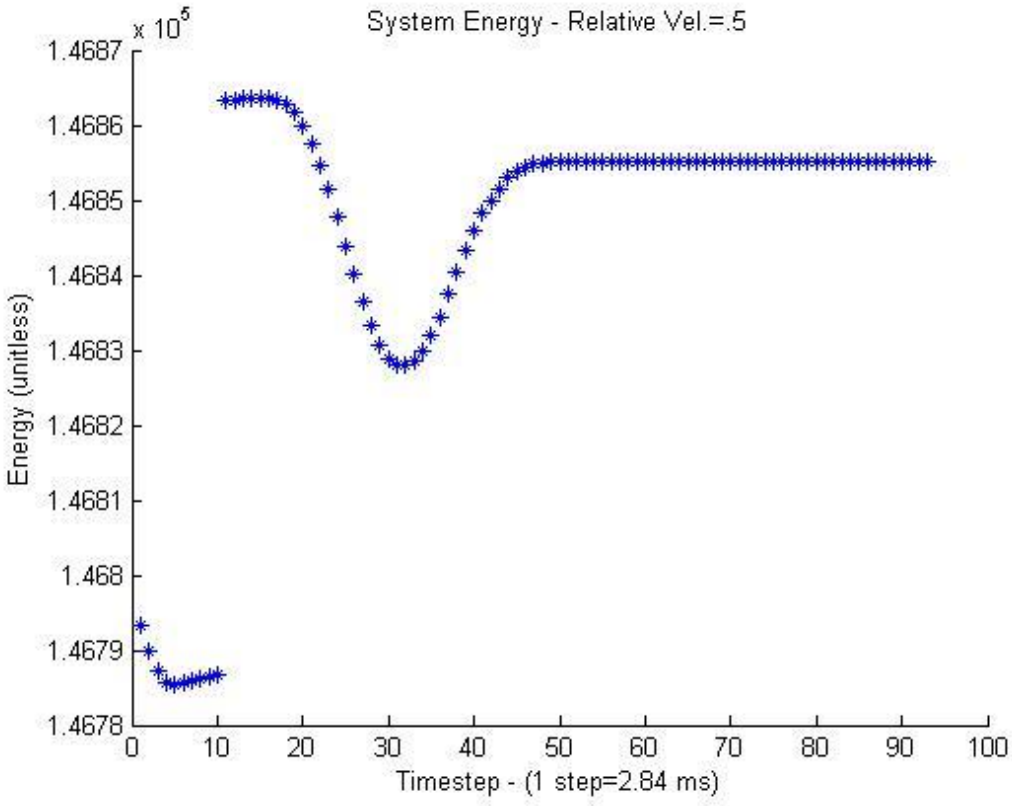

Figure 46: Total system energy of *K_s*=10 direct collision simulation.

The most important thing to note from the energy results is that the energies do not dramatically change over the course of the simulation. The simulations satisfy the conservation of energy because the equilibrium energy drops as a result of the collision. Examining the velocities of the system shows the cause of the decrease in the total energy.

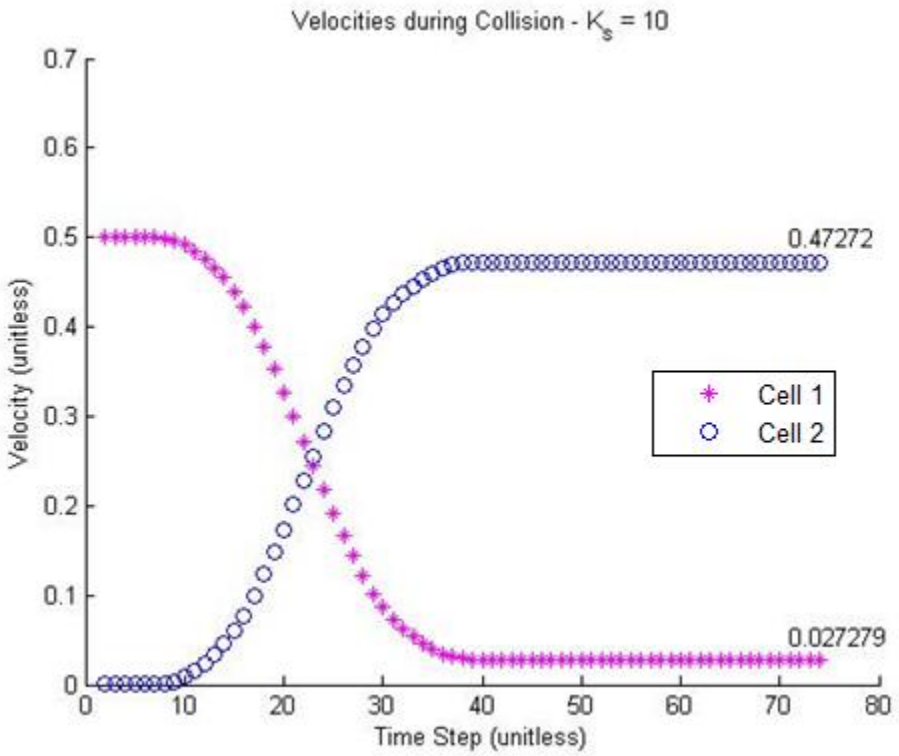

Figure 47: Total velocity of the particle's centers of mass throughout the simulation. The final velocity magnitude is shown on the graph for use later in this chapter.

This collision is very close to being a perfectly elastic collision where the kinetic energy is transferred entirely from one body to another. The total kinetic energy does clearly drop, however, while the different potential energy values stay roughly the same before and after the collision. This energy is lost due to the membrane damping, so energy conservation is satisfied in this simulation.

In continuing the experiment of varying *K_s*, the *K_s* value was increased to 75 and the same simulation was replicated to continue to analyze the corresponding data as shown in Figure 48.

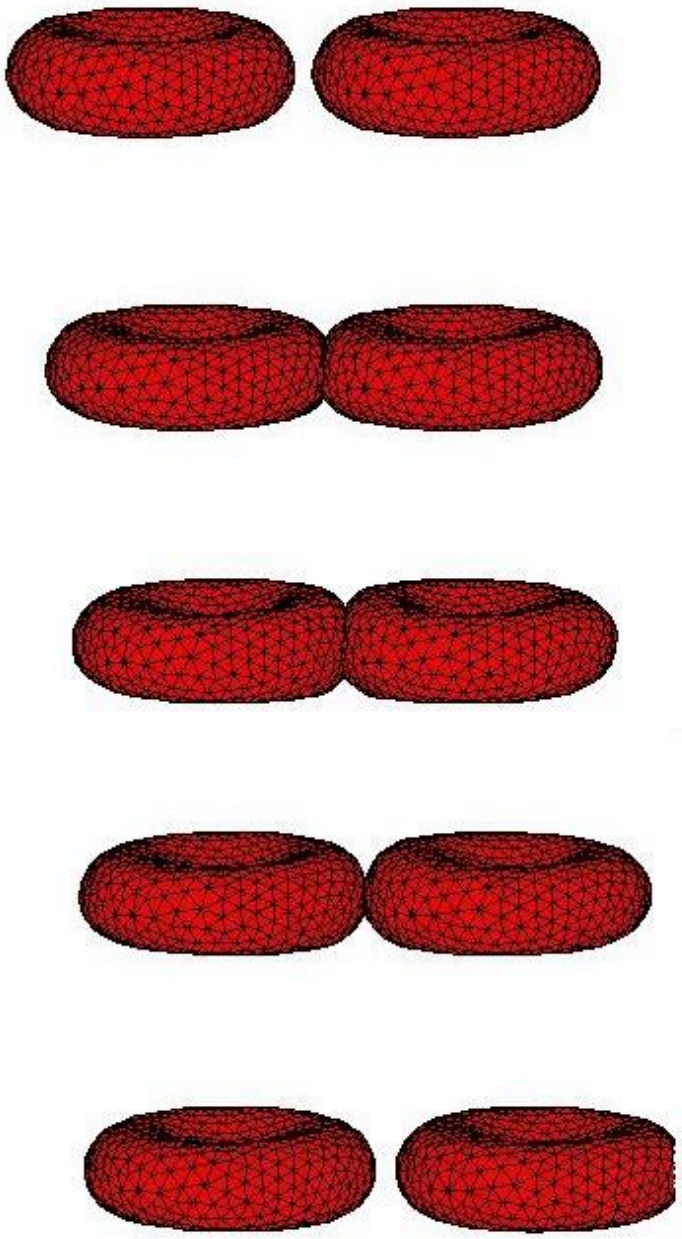

Figure 48: Collision progression of two red blood cells with *K_s*=75. Each progression represents Δt=28.4ms.

After *K_s* was increased, it can be seen that local deformation after contact increases significantly. The noticeable "bulge" at the contact point is much more apparent in this simulation compared to the first.

The energy of this system (Figure 49-Figure 51) can be similarly analyzed as it was for *K_s*=10.

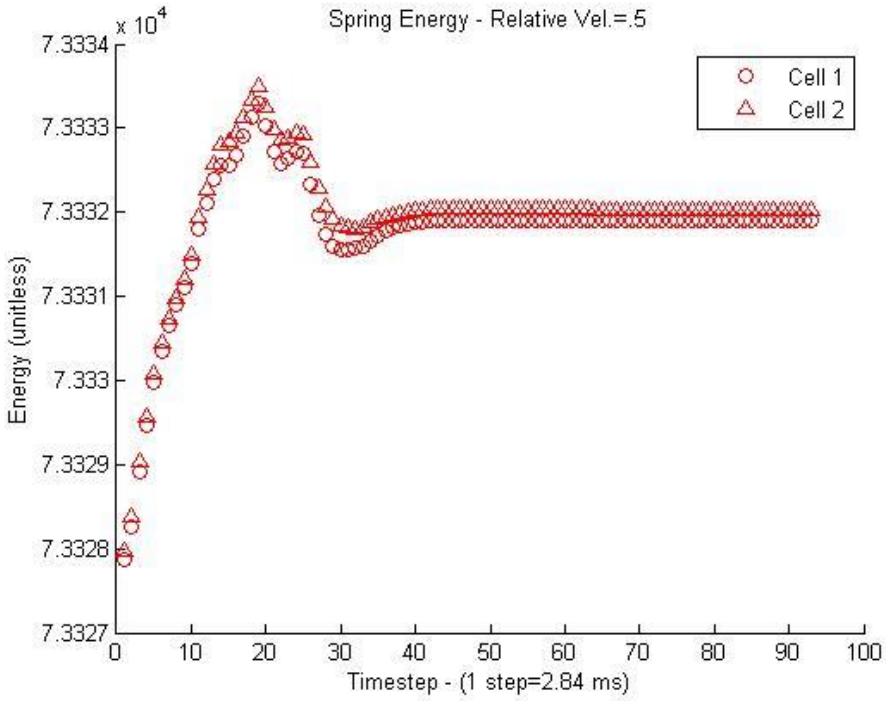

Figure 49: Spring potential energy of *K_s*=75 direct collision simulation.

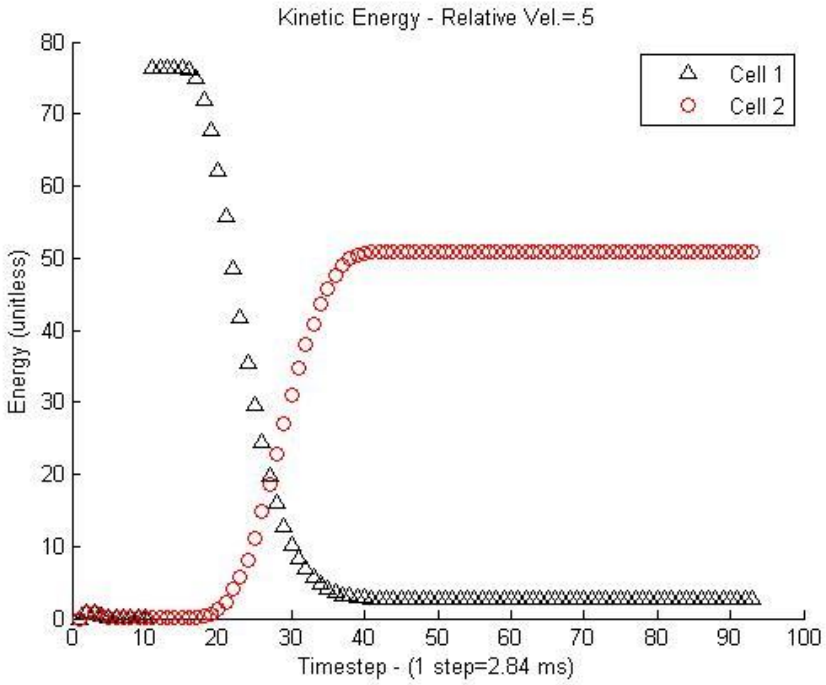

Figure 50: Kinetic energy of *K_s*=75 direct collision simulation.

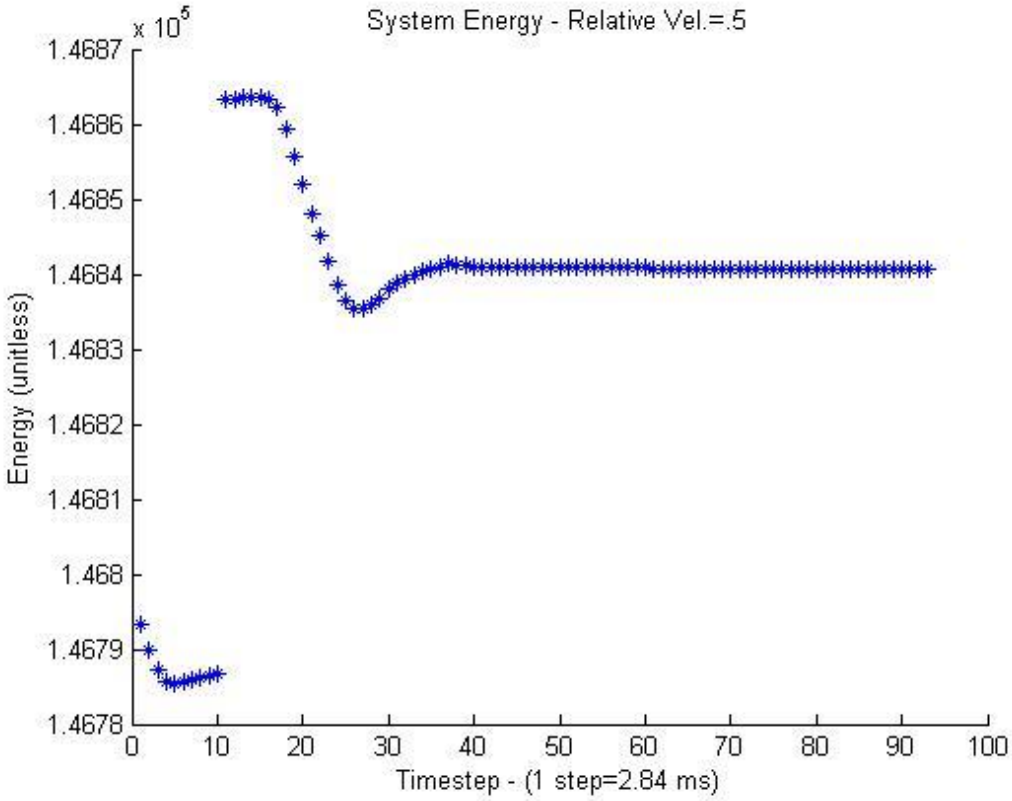

Figure 51: Total system energy of *K_s*=75 direct collision simulation.

As would be expected, the energy fluctuations from the simulation with a higher *K_s*=75 are higher than the energy fluctuations from *K_s*=10. The *K_s*=75 simulation was run as an intermediate simulation in a range where the resulting physics appear most correct. Further simulations were carried out to calculate the effective range of acceptable *K_s* range which will allow for the analysis and presentation of data that can be used for experimental verification.

In continuing to test the *K_s* range, *K_s*=100 was selected as the next test (Figure 52).

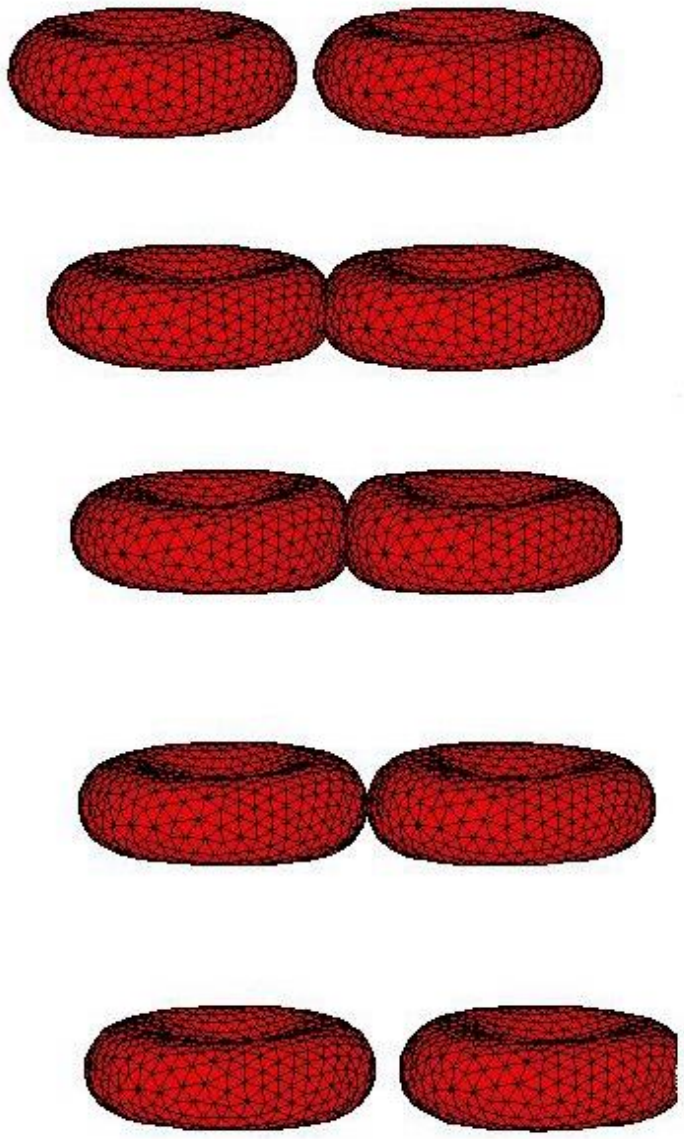

Figure 52: Collision progression of two red blood cells with *K_s*=100. Each progression represents Δt=28.4ms.

A trend can now be seen for the relationship between the local deformation of the cell and the effective spring constant applied to the intersected nodes. This makes good sense because the higher spring constant linearly increases the force applied to each

node. A check of the energies should be conducted again to ensure that there are no undesirable fluctuations and are shown in Figure 53-Figure 56.

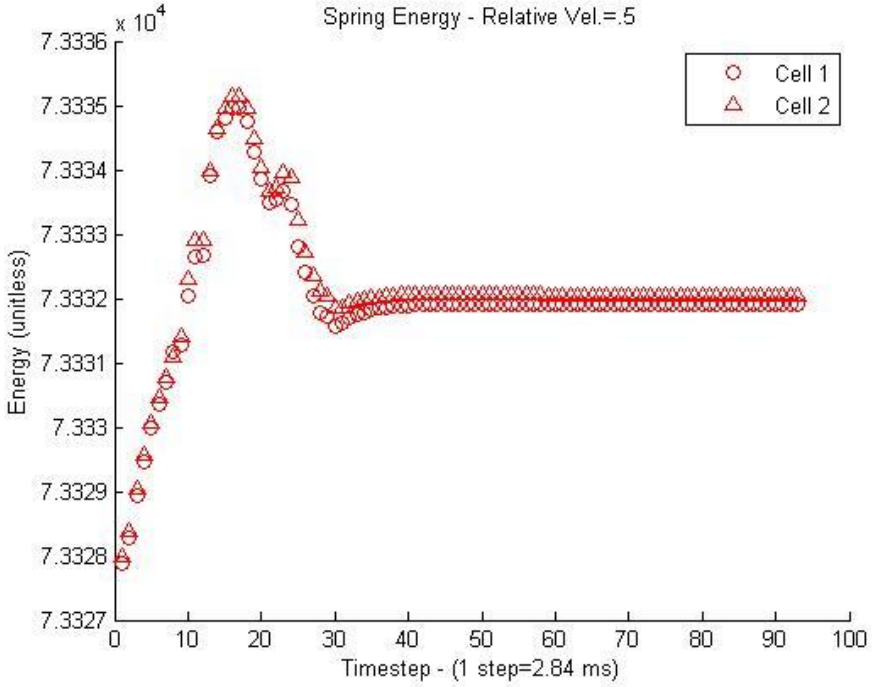

Figure 53: Spring potential energy of *K_s*=100 direct collision simulation.

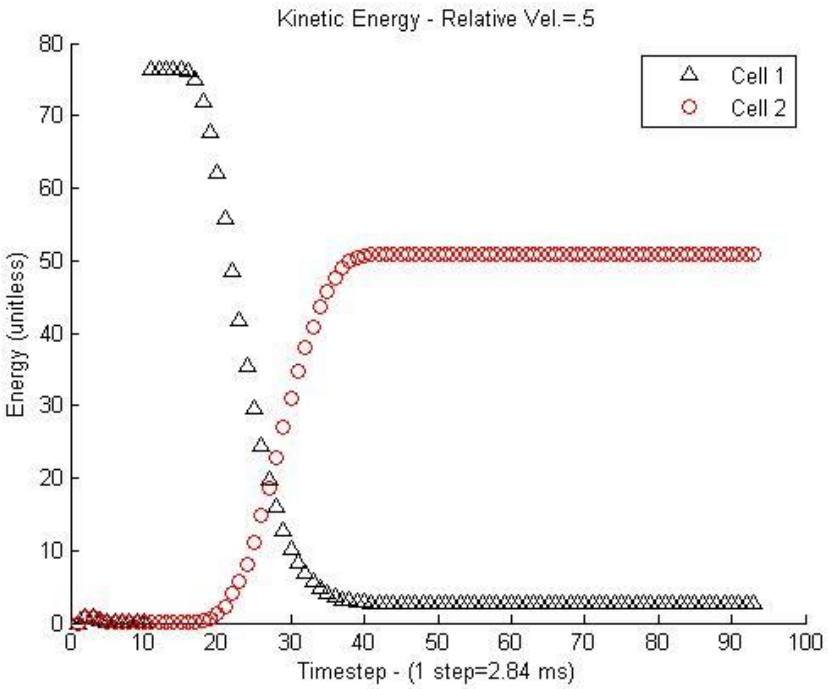

Figure 54: Kinetic energy of *K_s*=100 direct collision simulation.

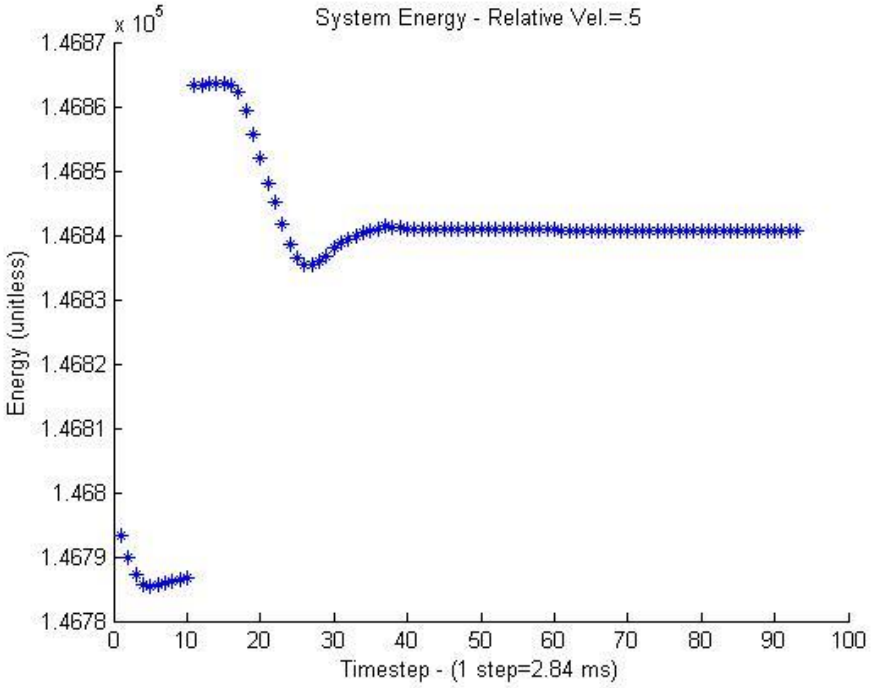

Figure 55: Total system energy of *K_s*=100 direct collision simulation.

Tracking the spring potential energy from the *K_s*=100 simulation heeds smooth potential energy terms with no major jumps that would suggest instability.

Examining the final velocities of the entire cell (see Figure 56) reveals an interesting phenomenon when compared to the original *K_s*=10 simulation. How the velocities change based on *K_s* will be further explained in the subsection following the presentation of data.

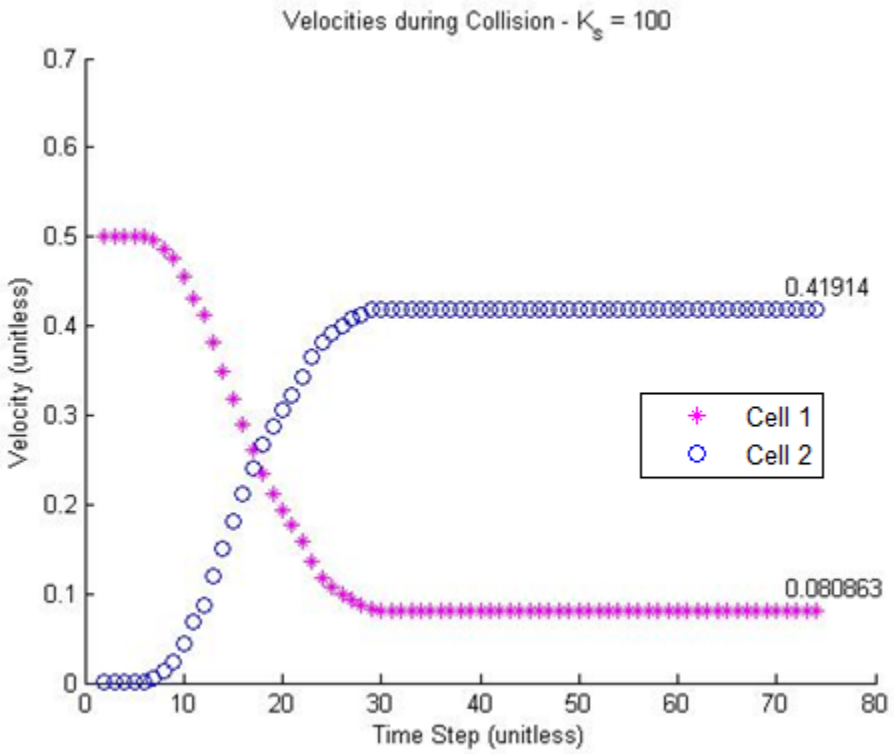

Figure 56: Total velocity of the particle's centers of mass throughout the simulation *K_s*=100. The final velocity magnitude is shown on the graph for use later in this chapter.

The *K_s* can continue to be increased until we can find a level of instability which will allow us to propose a potential range for *K_s* values to be checked against experimental data. Simulations were carried out for *K_s*=200 to find this outer limit for reasonable red blood cell response and shown in Figure 57-Figure 60.

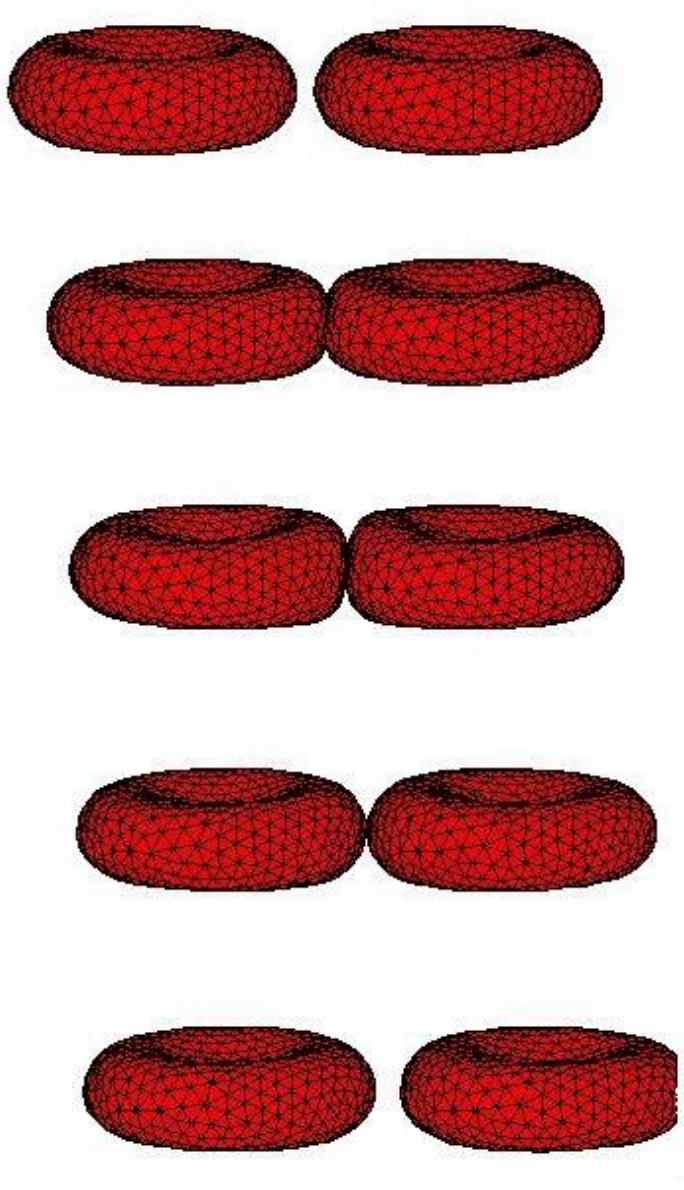

Figure 57: Collision progression of two red blood cells with *K_s*=200. Each progression represents Δt=28.4ms.

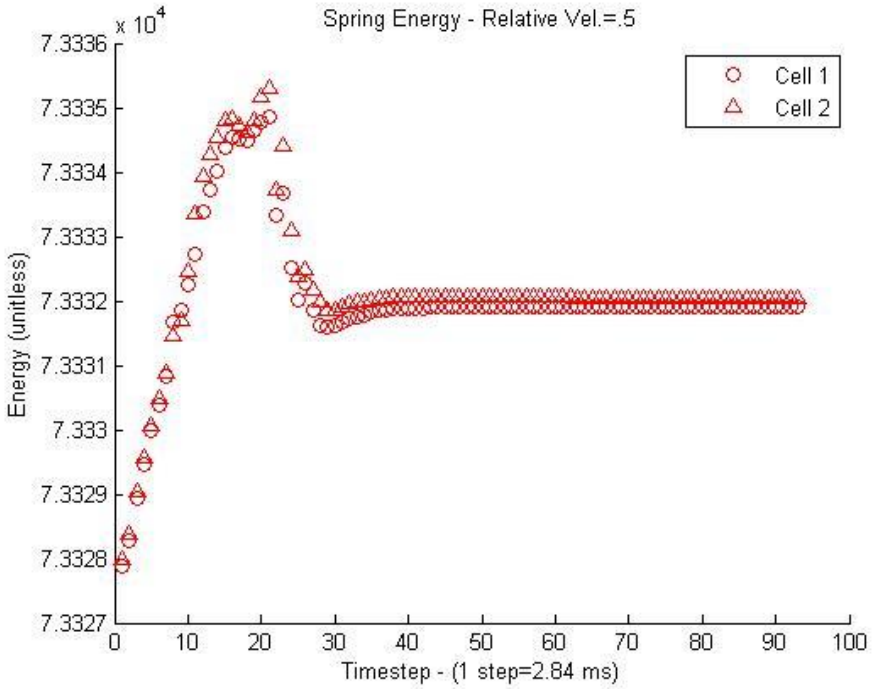

Figure 58: Spring potential energy of *K_s*=200 direct collision simulation.

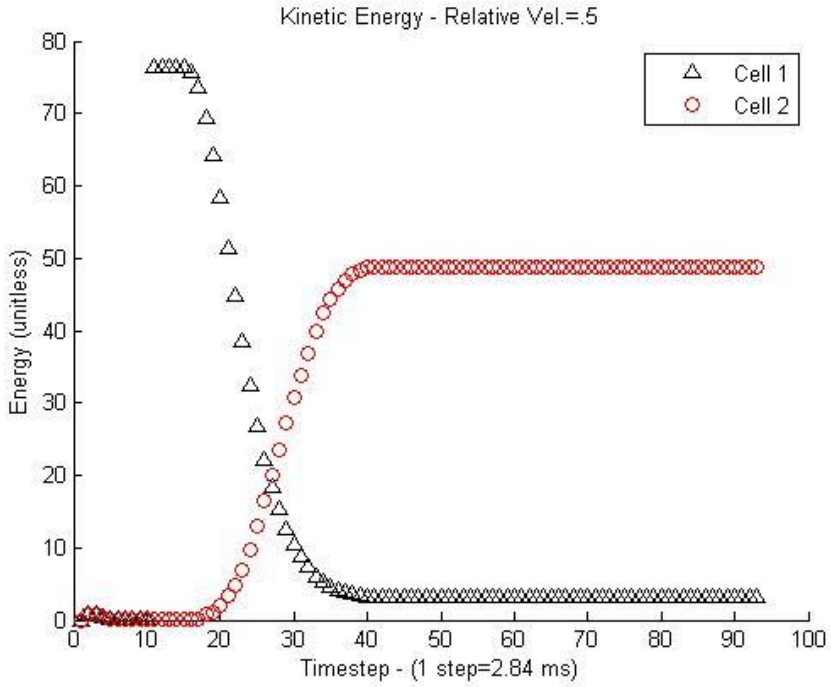

Figure 59: Kinetic energy of *K_s*=200 direct collision simulation.

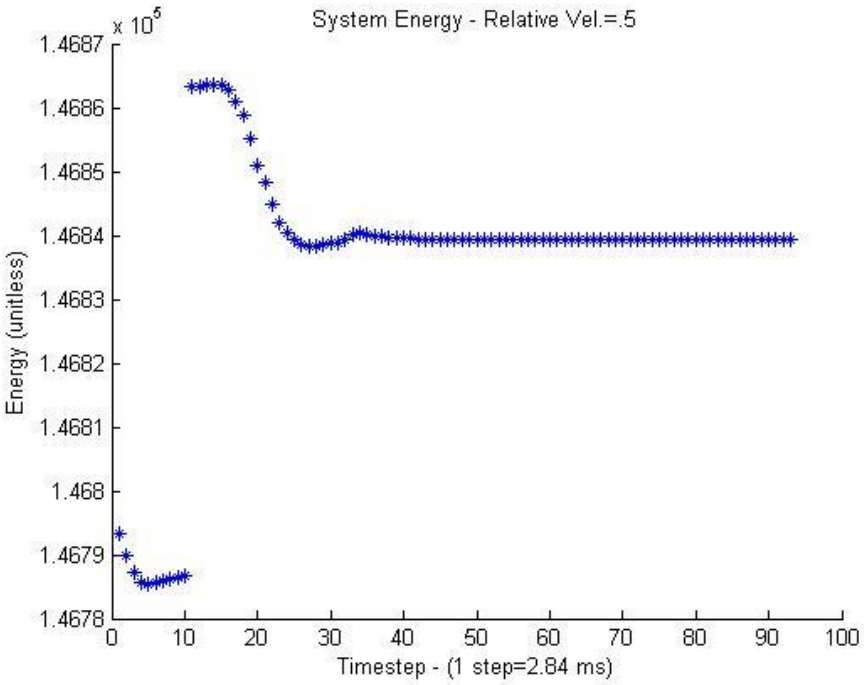

Figure 60: Total system energy of *K_s*=200 direct collision simulation.

The shape evolution of the *K_s*=200 simulation follows the expected pattern in that the local deformation is increased and the penetration depth is decreased. More importantly, there is a growing level of instability in the bending potential energy term as the force exerted becomes more sudden. The bending potential energy terms are mostly stable, but it can be seen that there is a small portion of the simulation where the energy is approaching a level of discontinuity that would suggest instability in the simulations. Increasing the *K_s* value again allows us to test further the level of appropriate *K_s* values as shown in Figure 61-Figure 64.

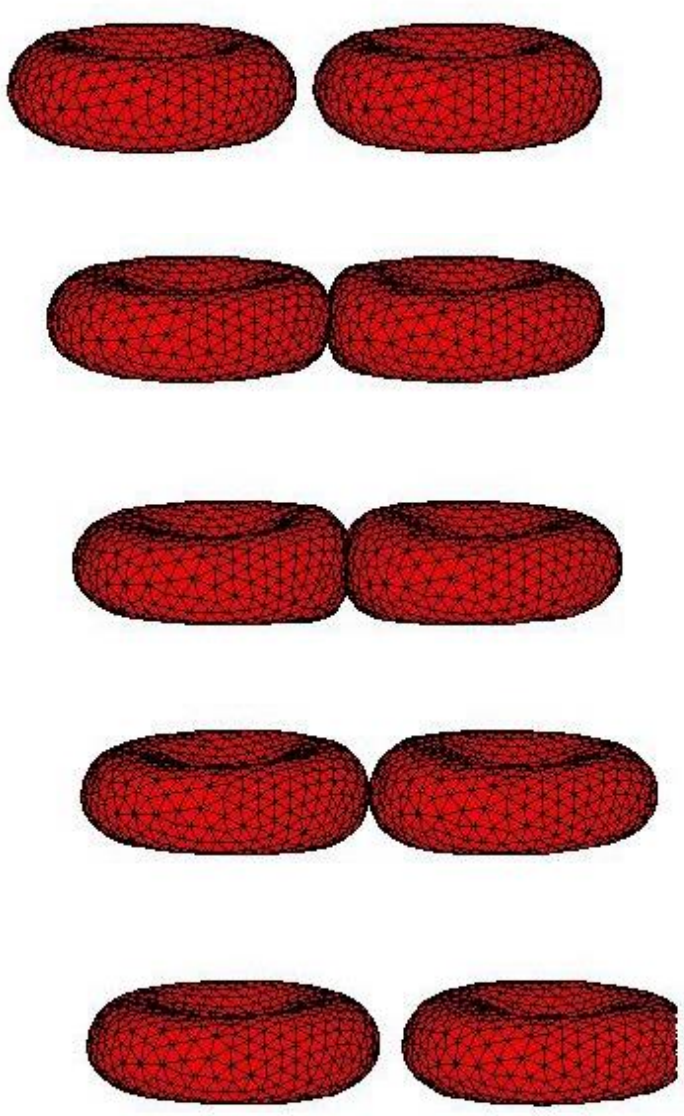

Figure 61: Collision progression of two red blood cells with *K_s*=300. Each progression represents Δt=28.4ms.

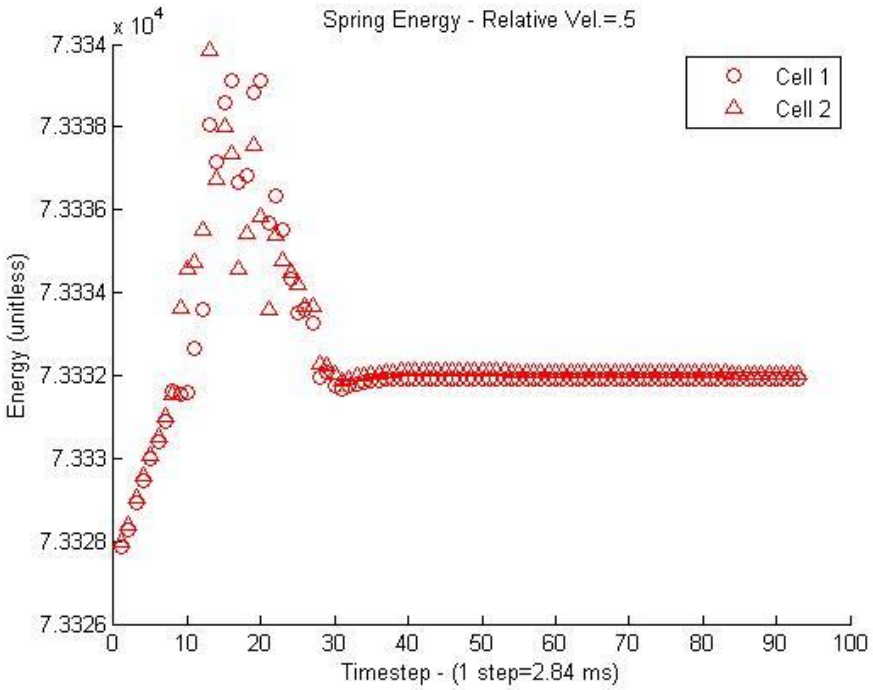

Figure 62: Spring potential energy of *K_s*=300 direct collision simulation.

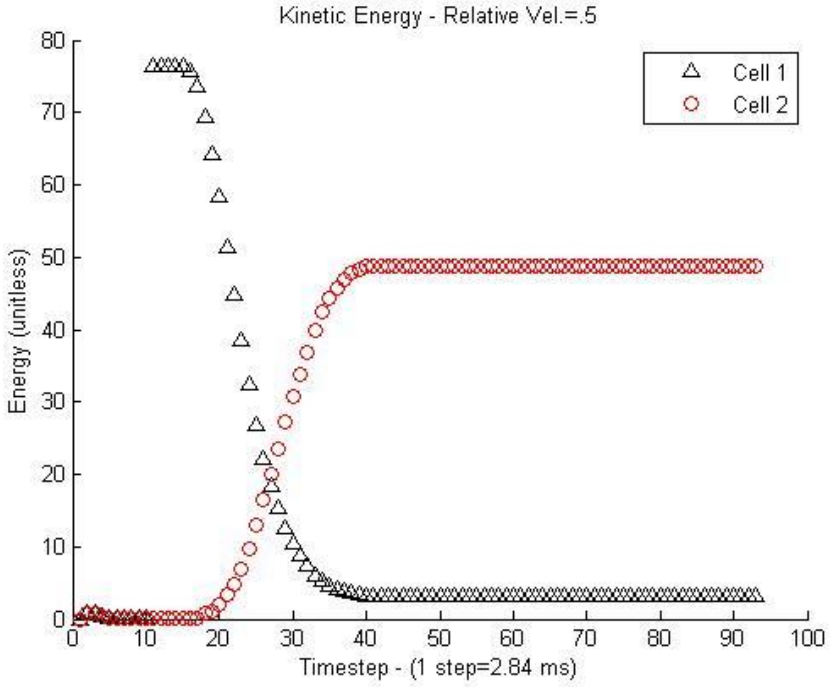

Figure 63: Kinetic energy of *K_s*=300 direct collision simulation.

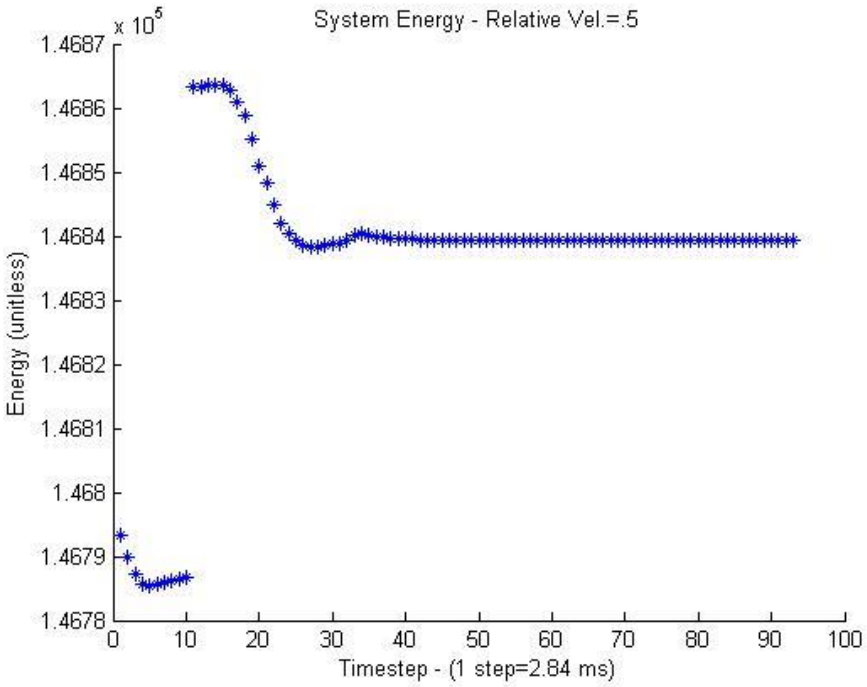

Figure 64: Total system energy for the K_s=300 direct collision simulation.

While the shape progression shown for the *K_s*=300 simulation appears to be as feasible as the simulations prior, the red blood cells, upon collision, actually undergo an unrealistic rippling and repulsive effect.

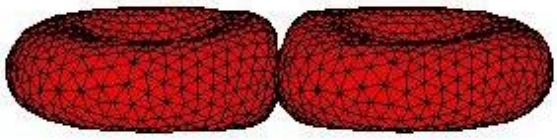

Figure 65: Visualization of the red blood cells between the second and third time step shown above in the shape evolution for *K_s*=300. It can clearly be seen that the red blood cells have lost contact with one another despite the relative velocity of the particles indicating that they are still moving toward one another.

From Figure 65 above, it is apparent that the red blood cells have lost contact with one another mid-collision. The above figure corresponds to halfway between the second and third steps from Figure 61. Because the cells are in contact during the second and third time step from said figure, it cannot be that the cells lose contact

without an external force. There are no external forces applied to these cells once they begin moving, so the results pictured above cannot be physically accurate. This would seem to indicate that the collision force is too strong for the red blood cell model to be able to compensate and replicate the cell membrane appropriately.

There are also problems that arise from this excessive repulsion phenomenon with the energies. In examining the potential energy term above, the plot is largely discontinuous in that it does not create a smooth curve as was seen for lower *K_s* values. This appears to indicate unphysical motion which can be clearly seen in the figure above. Therefore, although the *K_s*=300 appears as though it may create potentially unphysical results for this specific test setup, it will be regarded as the uppermost limit on *K_s*. An effective limit can be placed on *K_s* such that *K_s* is less than 300, but greater than 10. This will serve as a reference for the following simulations.

It is interesting to compare the local deformation of each simulation to see how *K_s* affects the contact area. The following figure (Figure 66) will show the red blood cells at the same time step while varying *K_s*.

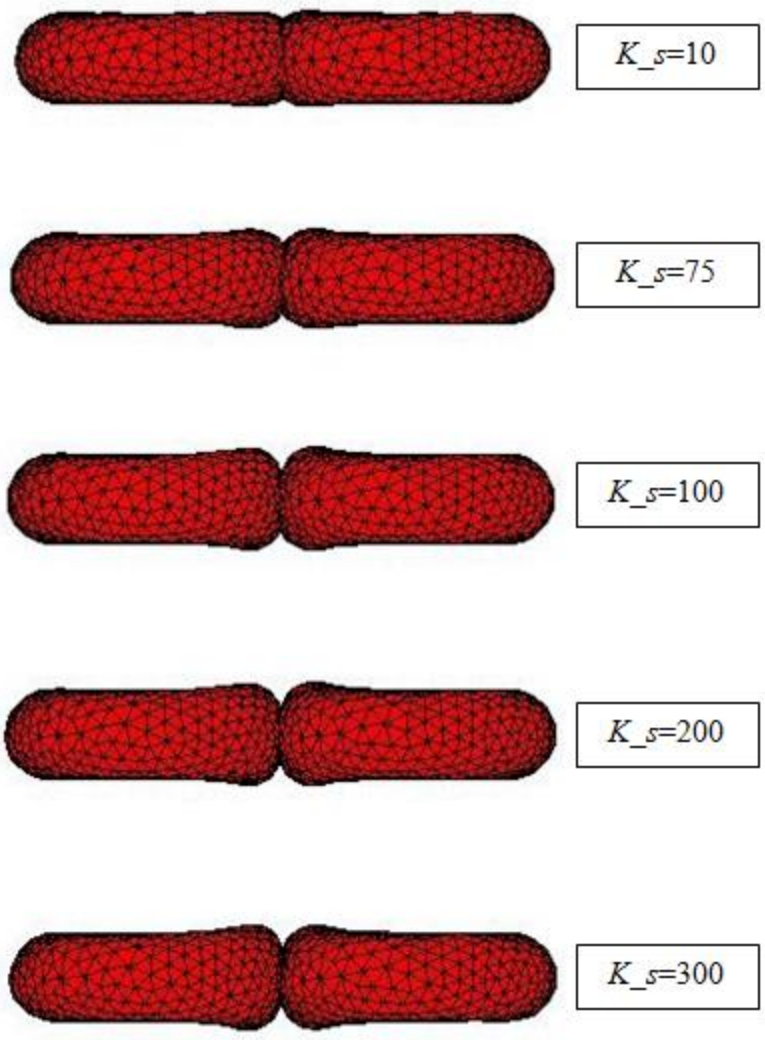

Figure 66: Local deformation of red blood cells at the same time step in the direct collision simulation.

K_s	Maximum Vertical Dimension	Maximum Vertical Dimension (μm)
10	2.8011	2.649
75	2.9314	2.772
100	3.0397	2.874
200	3.0323	2.867
300	3.2845	3.106

Table 2: Deformation measure in the minor radius direction from the simulations presented above. Values presented are dimensionless and microns, respectively.

The local deformation appears to be consistent after *K_s* passes a certain point. There are minor differences between *K_s*=75 and *K_s*=100 as would be expected but the differences between *K_s*=100 and *K_s*=200 are not apparent upon inspection. This observation is supported by Table 2. There is a clear upward trend of the amount of local deformation as the *K_s* value is increased. The effect is minimized at the larger values of *K_s*, namely between *K_s*=100 and *K_s*=200. The larger value when *K_s*=300 is not noteworthy because of the unphysical results mentioned earlier. This local deformation can serve as an effective way of validating experimental results. The dimensional values are given to facilitate such comparisons.

Section 5.1.2: Effective Coefficient of Restitution

One of the values that changed most upon changing *K_s* was the final velocity of each particle after collision. This change in velocity corresponded to a change in kinetic energy that differed based on the *K_s* value. The simulations in which the initially mobile cell transferred more of its velocity to the stationary cell tended to correspond to the softer *K_s* values, which would seem to indicate that these collisions are closer to elastic than those for higher *K_s* values. This makes physical sense because the larger *K_s* values lead to greater local deformation which increases the amount of membrane damping on the cell. The amount of membrane damping that a cell undergoes is directly related to the amount of kinetic energy lost because it was already shown that the four potentials on the cell return to equilibrium values after deformation.

This section will be used to evaluate the effect of changing the *K_s* value has on this kinetic energy transfer. For the purposes of this thesis, this transfer will be characterized by its effective coefficient of restitution. The effective coefficient of restitution will be defined as:

$$ECOR = \frac{v_1^f - v_2^f}{v_1^i - v_2^i} \tag{5.1}$$

This term is differentiated from the coefficient of restitution because the collisions being presented in this thesis are not instantaneous as is the case in typical applications of the coefficient of restitution. Event driven models use a true coefficient of restitution in their simulations as the velocity is explicitly controlled on impact, while the effective coefficient of restitution is a passive quantity that arises from the membrane damping defined earlier. The velocity profiles of the cells' centers of mass (Figure 67-Figure 71 are used because they are relatively unaffected by the local deformation at the collision which gives a smooth velocity profile throughout the simulations.

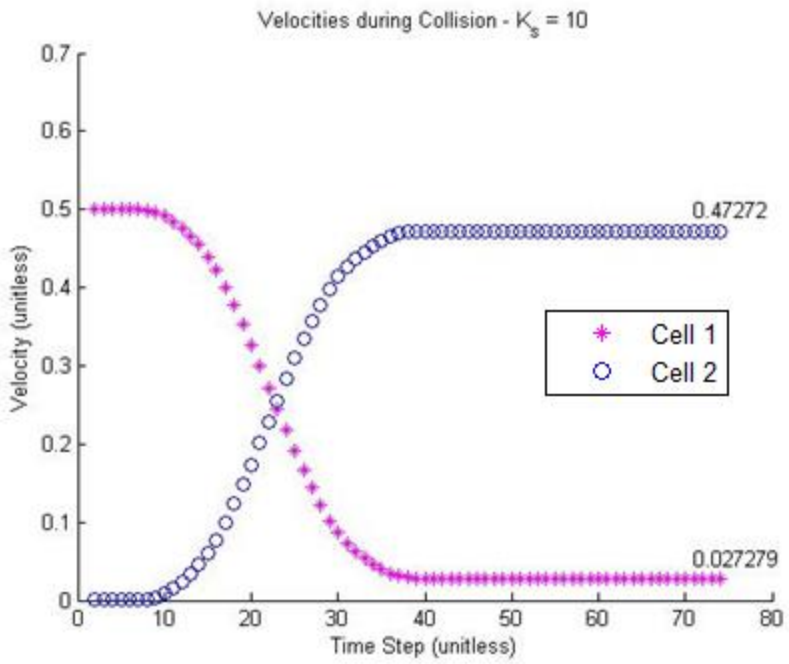

Figure 67: Velocity profile during simulations using *K_s*=10.

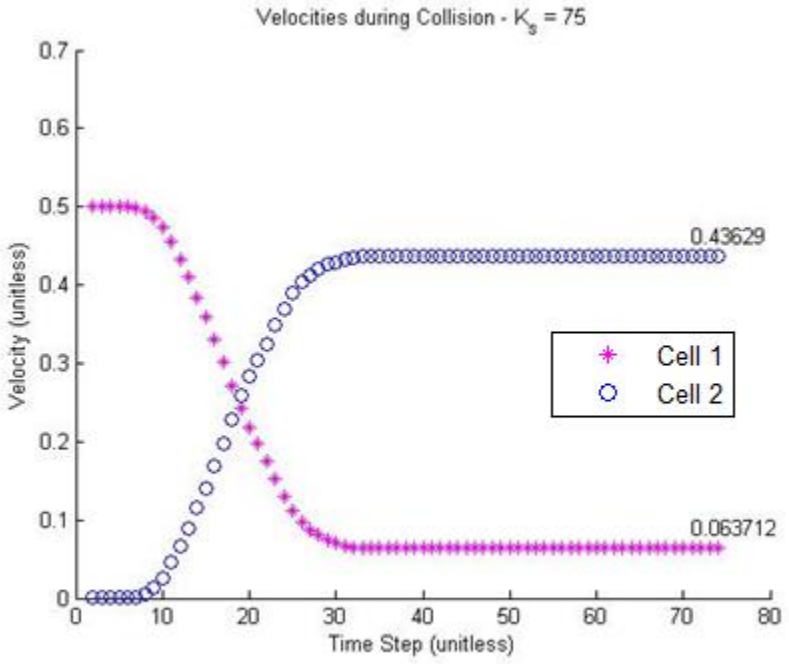

Figure 68: Velocity profile during simulations using *K_s*=75.

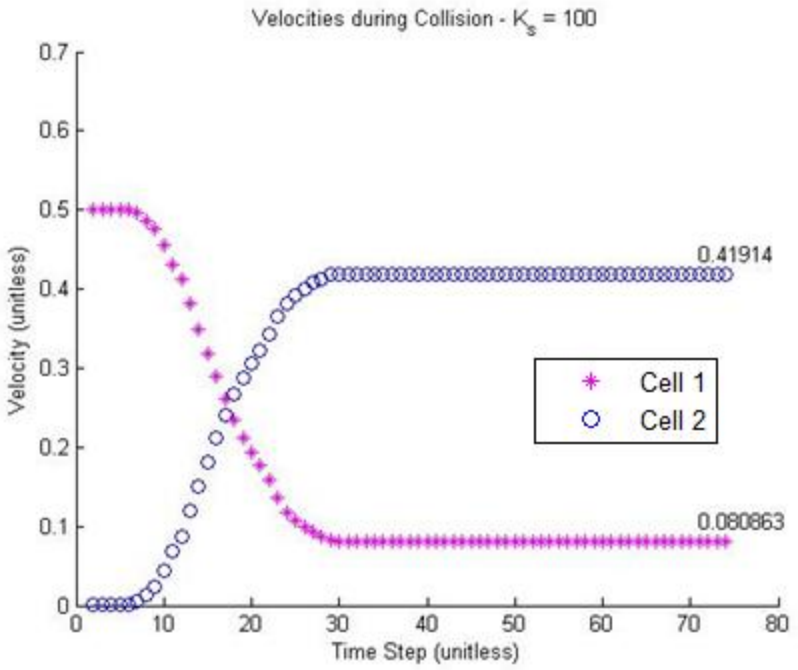

Figure 69: Velocity profile during simulations using *K_s*=100.

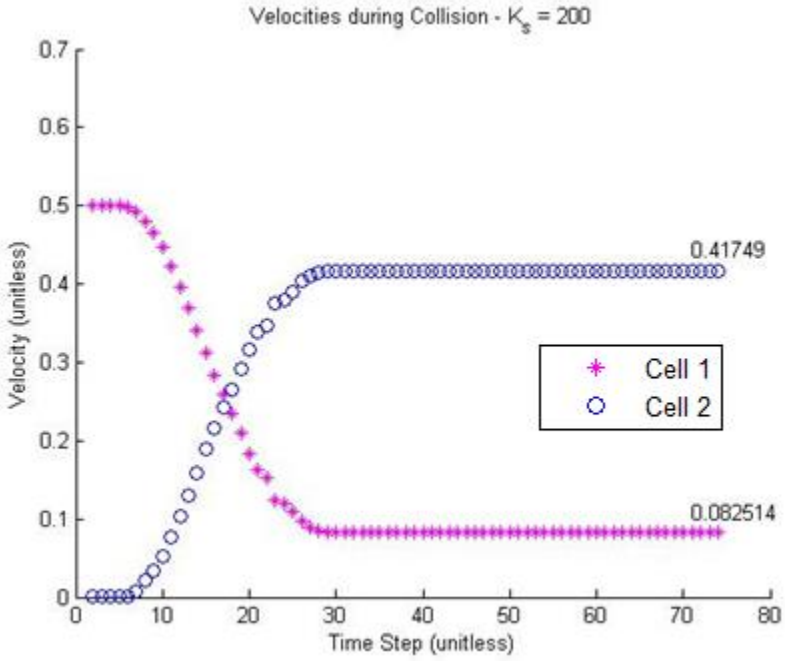

Figure 70: Velocity profile during simulations using *K_s*=200.

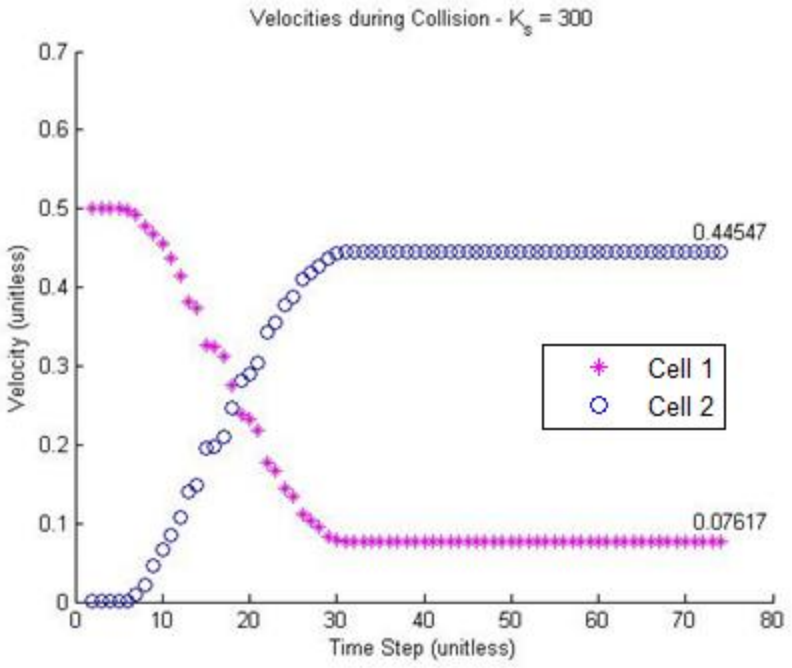

Figure 71: Velocity profile during simulations using *K_s*=300.

Using the equation shown above, we can calculate the effective coefficient of restitution range that can be achieved using this collision detection method.

K_s	ECOR
10	.8909
75	.7452
100	.6766
200	.6700
300	.7386

Table 2: Observed effective coefficient of restitution values for the direct collision simulation.

The above values are presented to be matched against experimental data that attempts to characterize a coefficient of restitution. At the time of writing this thesis, there is insufficient data to be able to appropriately match the simulation results to

experimental data. For this reason, the full range of effective coefficients of restitution is given with the hope that it will facilitate experimental verification.

The *K_s*=300 experiment appears to raise the effective coefficient of restitution after the value consistently falls as the *K_s* value increases. However, in examining the velocity profiles of the *K_s*=300 simulation, it should be noted that the velocities are actually unphysical in the context of the simulation. The momentum of the total system is not conserved because the sum of final velocities is greater than that of the initial velocities, implying that the total system momentum increases as a result of the collision. This clearly cannot be the case, so based on the table above, the effective coefficient of restitution of the red blood cell collisions ranges between .8909 and .6700. These values provide easily obtained experimental data to allow for comparison and validation.

Section 5.2: Angled Collision

Additional test cases were created to be able to further test the collision method. The normal collision in the previous section made sense to provide the simplest results and give data that would be easily compared against experimental data. The direct collision is, however, an ideal case for a system in which there are an infinite number of collision orientations. Therefore, a second test case was devised in which the normal force would typically only account for certain aspects of the resulting motion due to collision. In placing one cell at a 90 degree angle, the desired effect can be achieved. This will allow for the analysis of the method with a situation in which normal force effects would seem to be minimized. The results presented in the following section will either highlight or downplay the necessity for including a frictional force.

Section 5.2.1: Energy Conservation and Dissipation

The following figures (Figure 72-Figure 75) show the results of the K_s = 10 simulations.

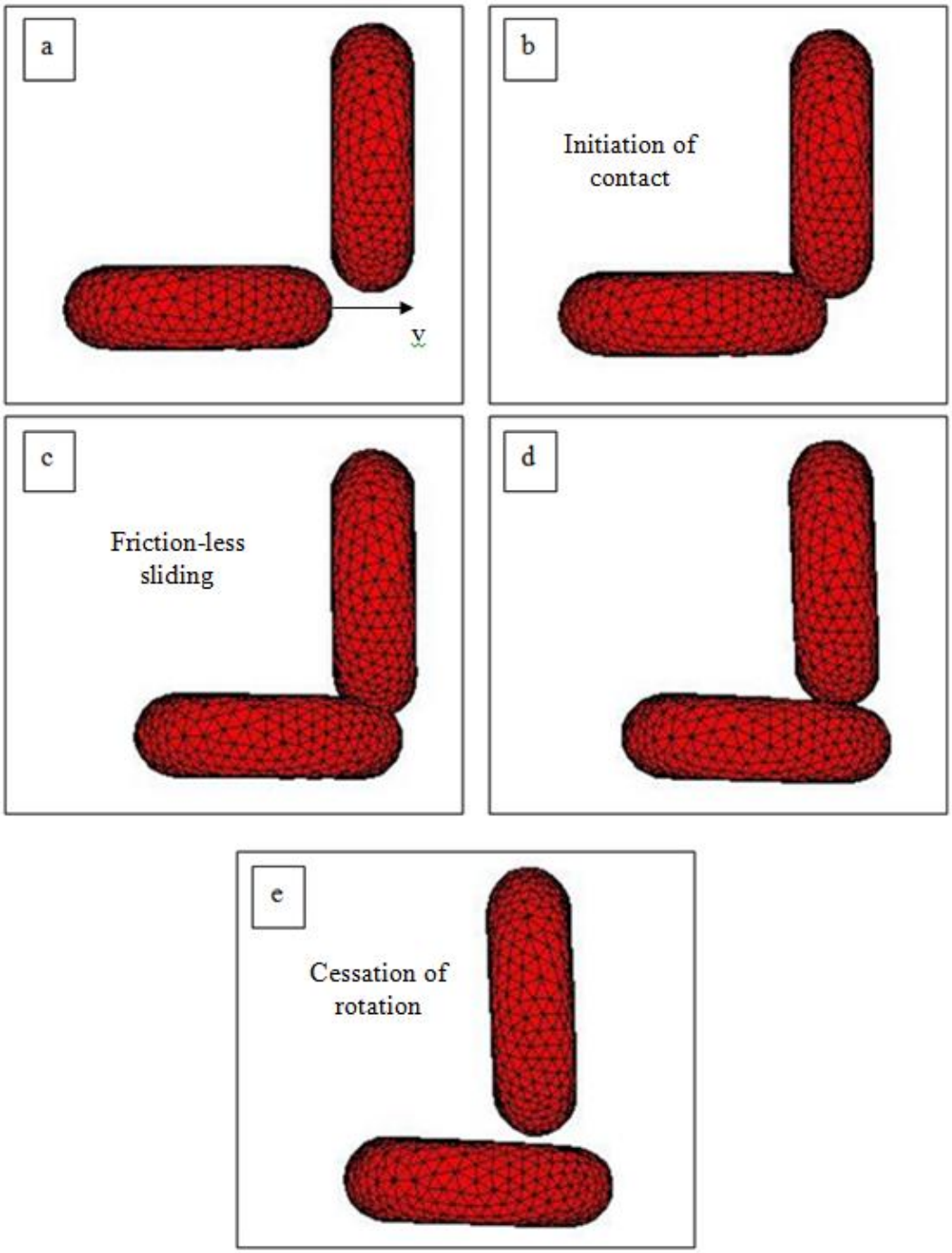

Figure 72: Shape evolution of the red blood cells in the rotated cell collision test case with K_s=10. The progression starts from the top left and moves to the right. Each progression corresponds to Δt=28.4ms.

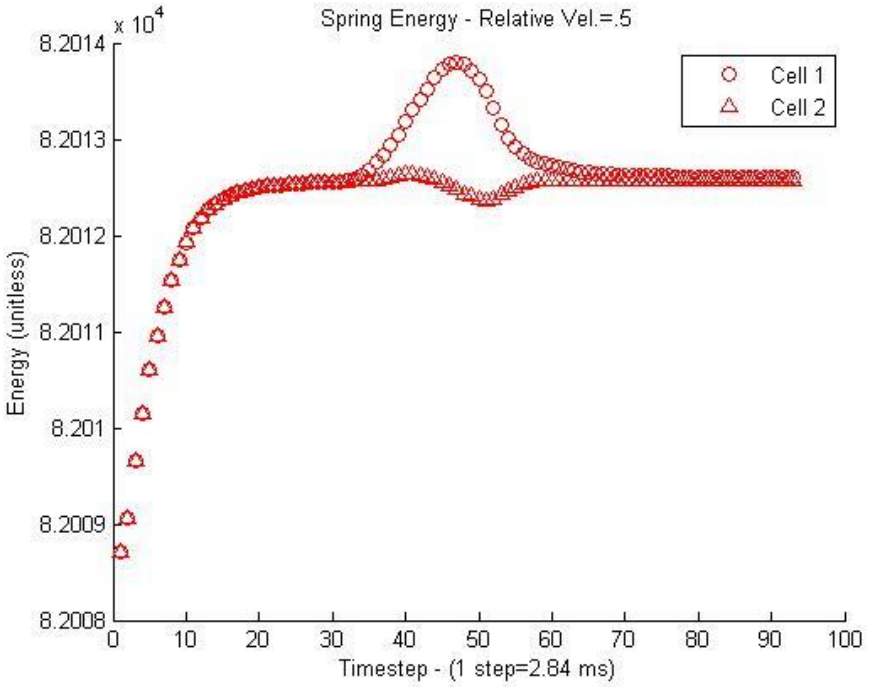

Figure 73: Spring potential energy of *K_s*=10 rotated collision simulation.

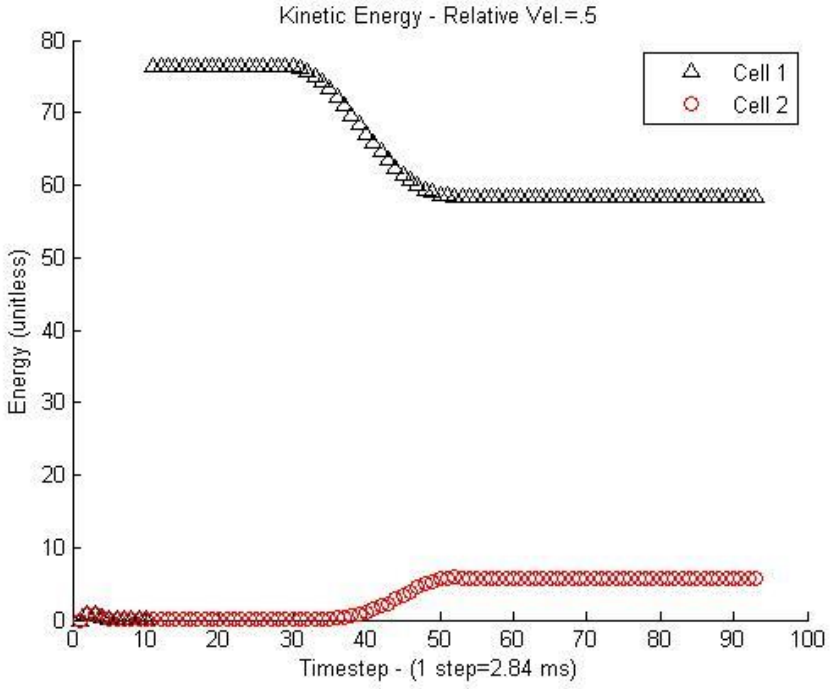

Figure 74: Kinetic energy of *K_s*=10 rotated collision simulation.

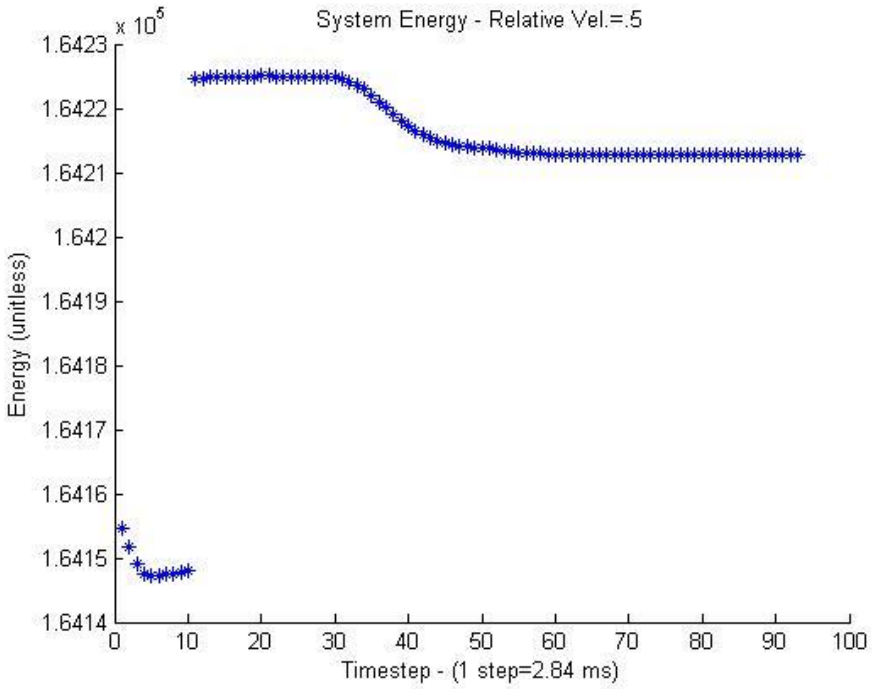

Figure 75: Total system energy of *K_s*=10 rotated collision simulation.

Before discussing the results, the definition of kinetic energy needs to be justified in this simulation. The red blood cell should have a total rotational energy because the cells are clearly rotating after impact. However, because of the discretized nature of the red blood cell, the calculation of this rotational energy is an abstract concept. The nodes are allowed to move with respect to one another which would indicate that the rotational velocity of the cell is not a single number, but would actually be different at every node. This issue is easily resolved by using the effective velocities as defined in Section 4.4.3. These velocities can be used to find the individual contribution of the nodes to the kinetic energy. As mentioned in Section 1.3, rotational effects of the individual nodes are ignored, so the total kinetic energy is defined as:

$$KE = \sum_{N_i} \frac{1}{2} m_i v_i^2 \tag{5.2}$$

As was the case in the direct collision section, we will be looking for both conservation of energy and continuity of the spring potential energy term as a criterion for stability in the simulations. These checks will be used to establish the range of *K_s* values that can be used for this specific situation. By checking collisions at 3 extreme conditions, a range of values that is well-suited for any condition can be projected for further simulations once experimental data becomes available for verification.

Figure 73 above suggests that the bending potential terms approach an equilibrium value as would be expected with a continuous curve, which would suggest stability. This stability is important in establishing the physical feasibility of the simulation with the values used. Second, in viewing the potential energy, it can be seen that any fluctuations are relatively minor. What these minor fluctuations imply is that there is no net change in the potential energy before and after the collision. There is, however, energy lost due to kinetic energy from membrane damping. The kinetic energy clearly decreases in the system before and after the system, so the net energy of the system goes down as would be expected in a collision of viscous bodies. From above, it can be concluded that *K_s*=10 presents a valid simulation. Smaller values of *K_s* should be avoided because the collision volume would then approach inappropriate levels that are typically considered to be unphysical because the larger a collision volume becomes, the larger the total local error in displacement.

In continuing this set of simulations, *K_s* was increased to 75 and the simulations were run again and displayed in Figure 76-Figure 79.

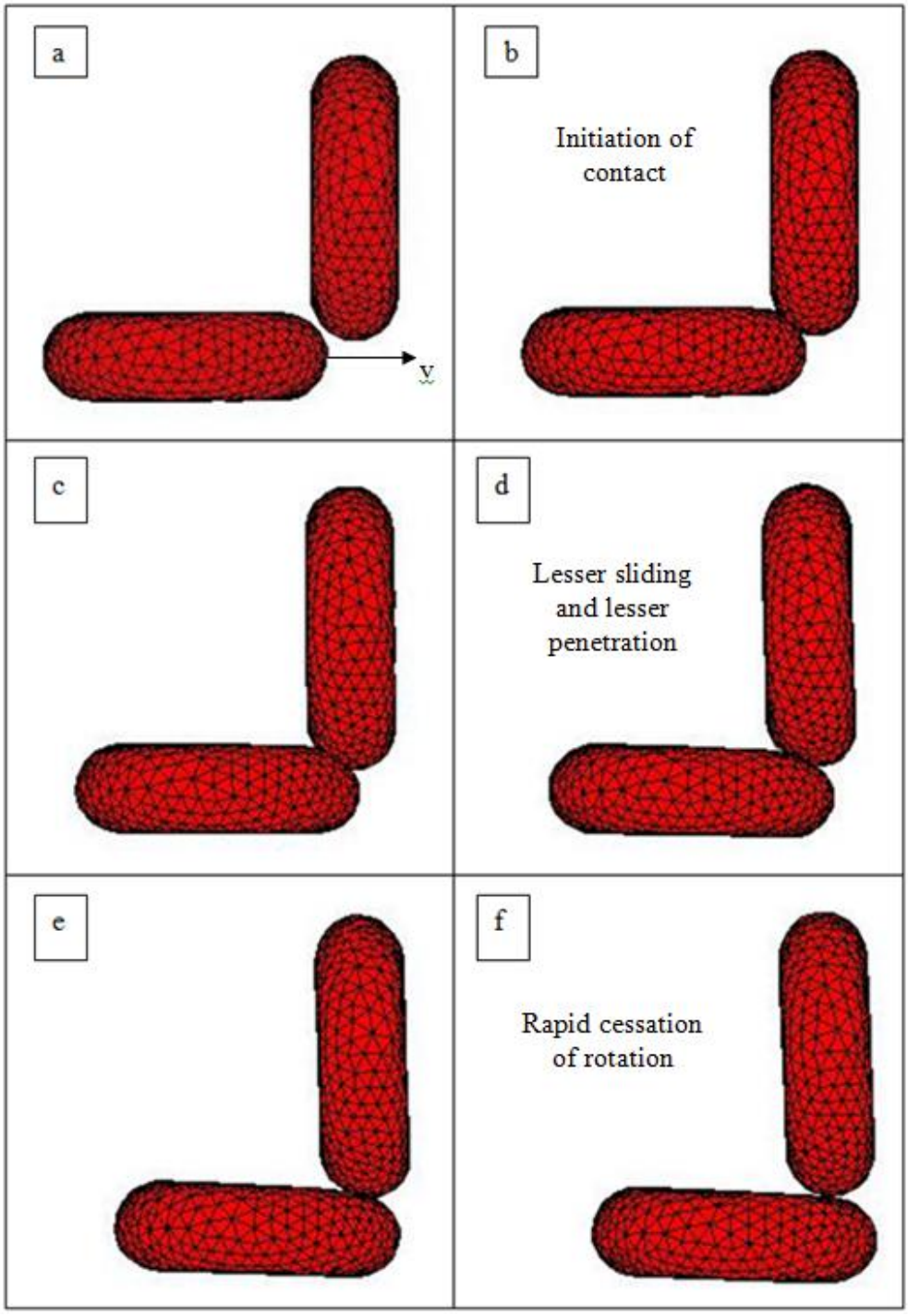

Figure 76: Shape evolution of the red blood cells in the rotated cell collision test case with *K_s*=75. The progression starts from the top left and moves to the right. Each progression corresponds to *Δt*=28.4ms.

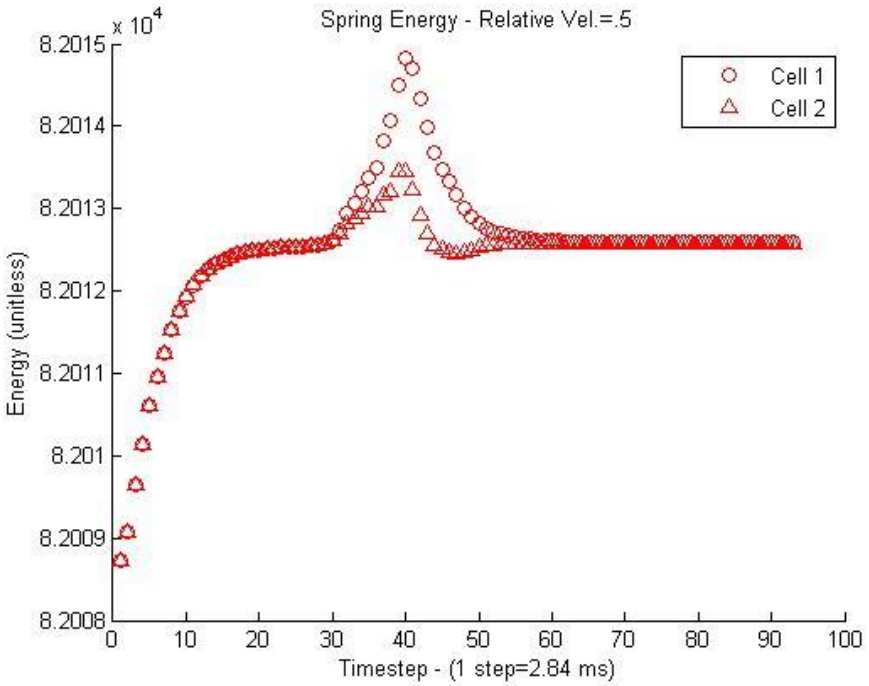

Figure 77: Spring potential energy of *K_s*=75 rotated collision simulation.

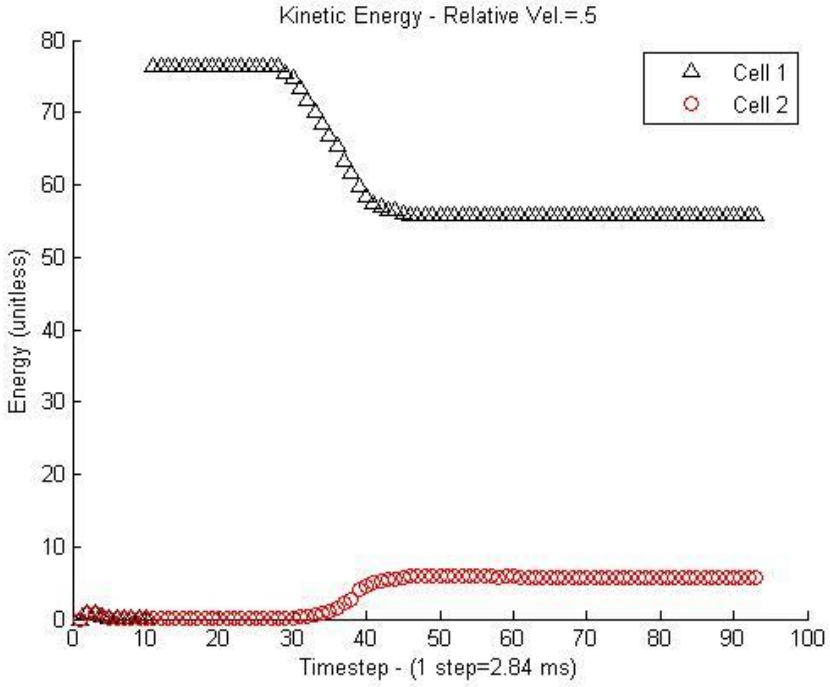

Figure 78: Kinetic energy of *K_s*=75 rotated collision simulation.

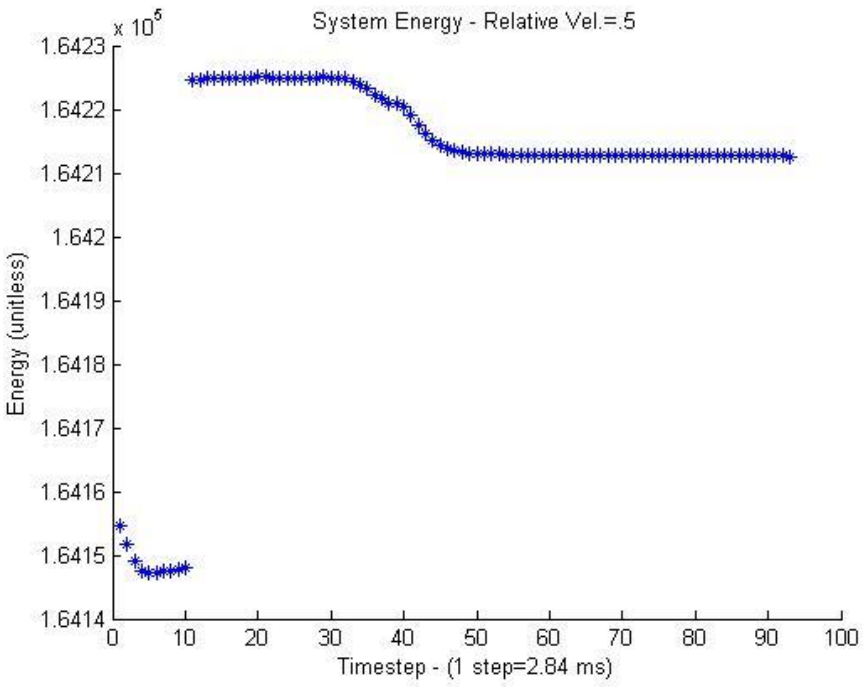

Figure 79: Total system energy of *K_s*=75 rotated collision simulation.

Note that the initial jump in energy is due to kinetic energy resulting from the initial velocity the first cell is given. This jump does not violate the conservation of energy because the cells are held fixed to allow for minimization initially. This means that the energy of the system at t=0 does not accurately represent the energy of the entire system. Instead, the energy of the system after the first particle begins moving is more appropriately considered the total system energy.

The *K_s*=75 simulations present data that is similar to *K_s*=10. The most notable difference is that of the change in local deformation which will be examined later in this section. However, the *K_s*=75 simulation results present themselves in a way that is very similar to *K_s*=10 with respect to energy conservation and continuity. Though the curves for spring potential energy are noticeably sharper upon initialization of the collision, the curves still present themselves as smooth. As was

the case in *K_s*=10, the energy lost in the kinetic energy due to membrane damping is significantly larger than the energy fluctuations seen in the potential energy. This is shown more clearly in that the system's total energy decreases through the simulation. From this, the conservation of energy is satisfied which would indicate that the results are realistic. The *K_s*=75 simulation seems to represent one that is very accurate in visualizing the results while also presenting the most stable potential energy terms. It is important to continue to find a maximum value of *K_s* such that we can present a full range *K_s* values that are appropriate for matching against experimental data. As such, *K_s* was increased to 100 and the simulation was conducted again (see Figure 80-Figure 83).

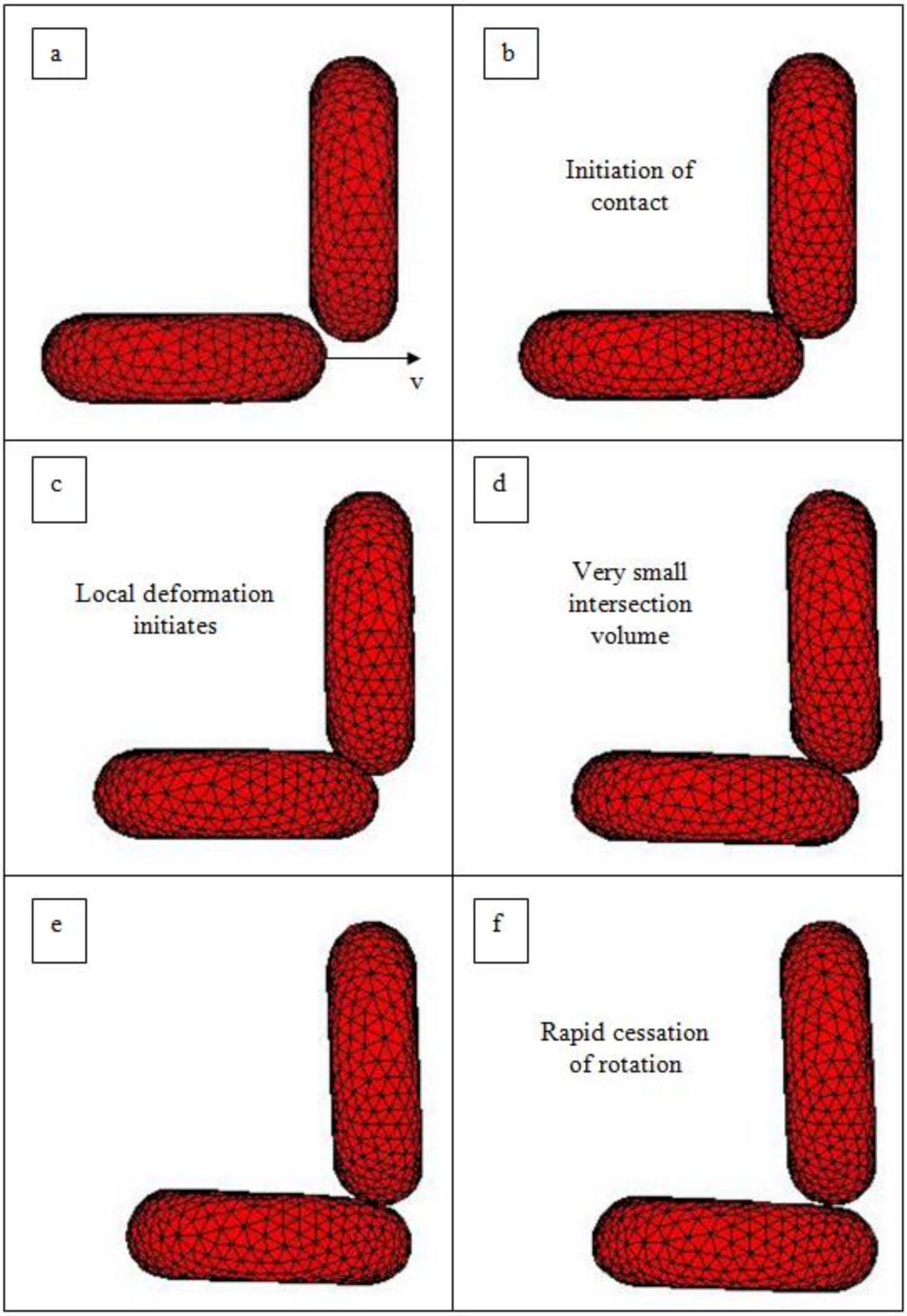

Figure 80: Shape evolution of the red blood cells in the rotated cell collision test case with *K_s*=100. The progression starts from the top left and moves to the right. Each progression corresponds to *Δt*=28.4ms.

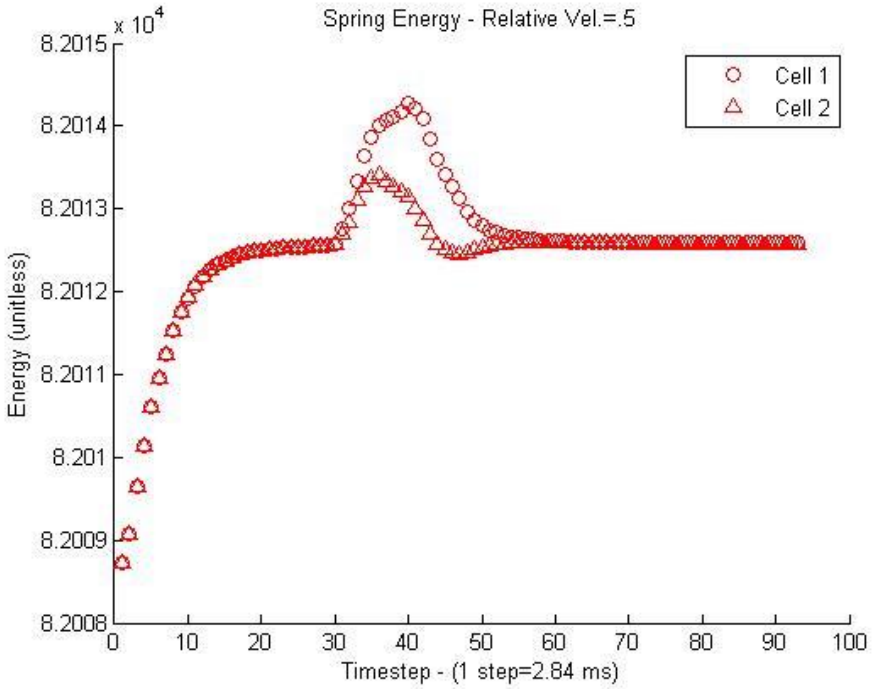

Figure 81: Spring potential energy of *K_s*=100 rotated collision simulation.

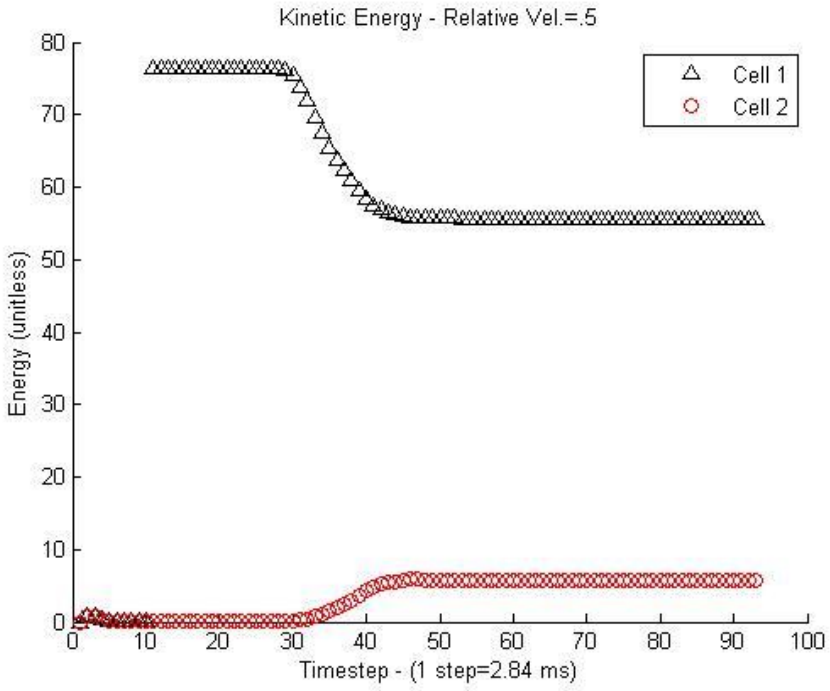

Figure 82: Kinetic energy of *K_s*=100 rotated collision simulation.

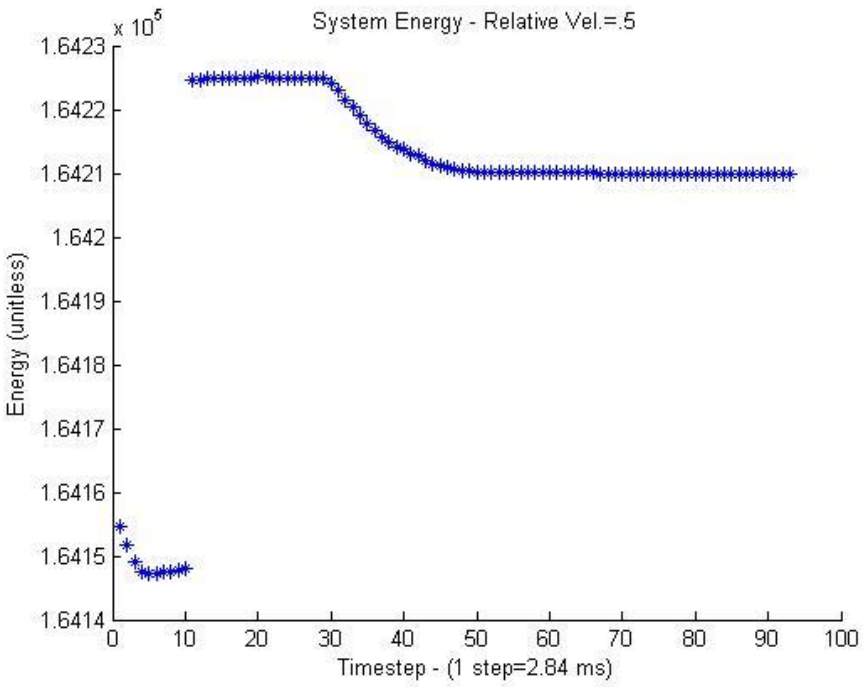

Figure 83: Total system energy of *K_s*=100 rotated collision simulation.

As would be expected, the results from the *K_s*=100 simulation are similar to the results from the *K_s*=75 simulation. One of the most notable differences is the change in local deformation that results from a higher *K_s*. The membrane becomes less penetrable as would be expected because of a large repulsion force upon collision. The difference is relatively minor which would seem to suggest that small changes in *K_s* do not result in significant changes, but large changes can create noticeable differences in the behavior of the system. As was the case with the *K_s*=75 simulation, the energy of the system follows a smooth, continuous curve which would suggest stability. In this respect, the simulation appears valid.

Also, in noticing that the kinetic energy undergoes a relatively large drop due to the membrane damping from the collision, the total energy of the system decreases which satisfies the conservation of energy because we expect the total energy of the system

to drop in inelastic collisions. For the *K_s*=100 simulations, Figure 84-Figure 87 are presented below.

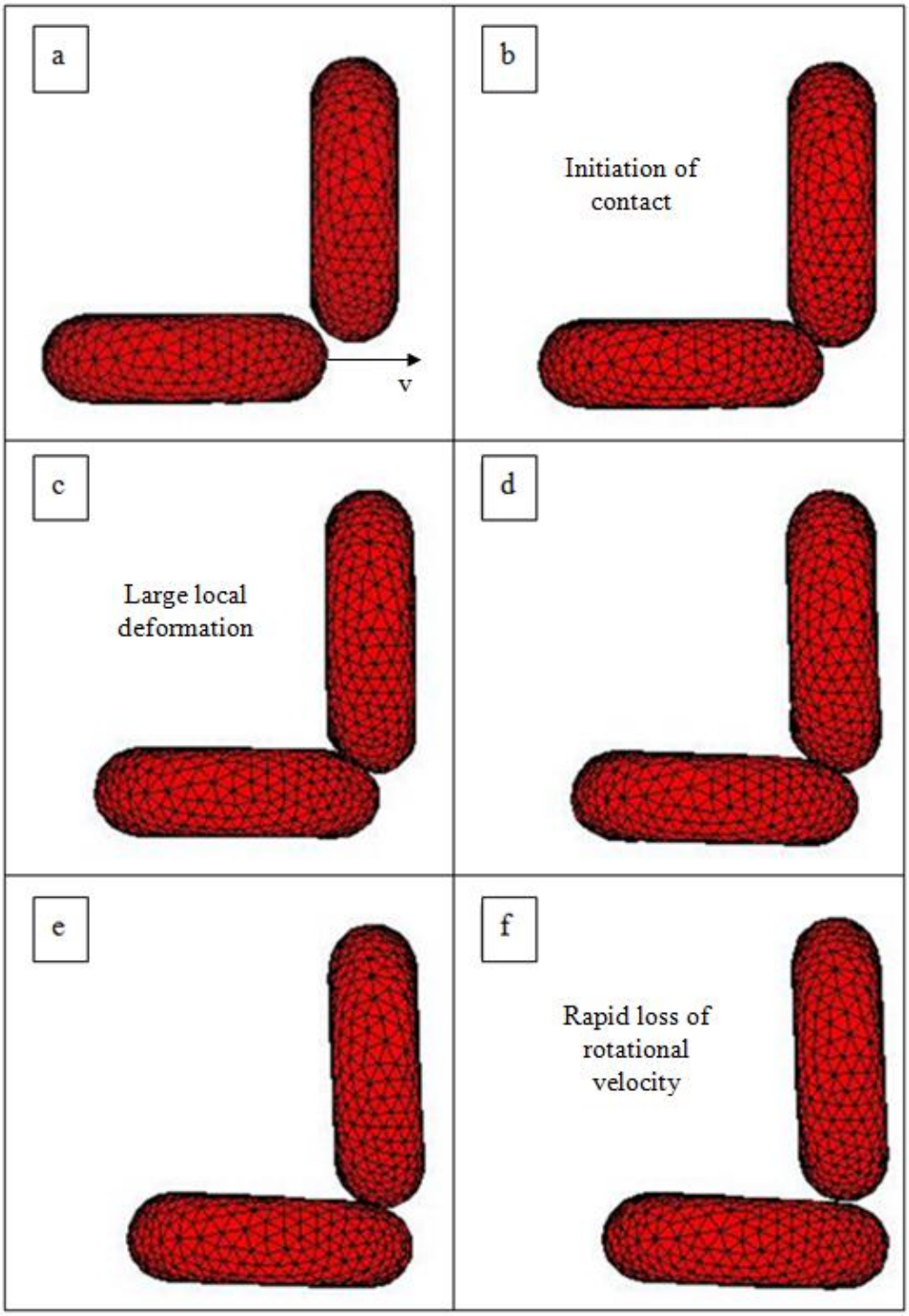

Figure 84: Shape evolution of the red blood cells in the rotated cell collision test case with *K_s*=200. The progression starts from the top left and moves to the right. Each progression corresponds to *Δt*=28.4ms.

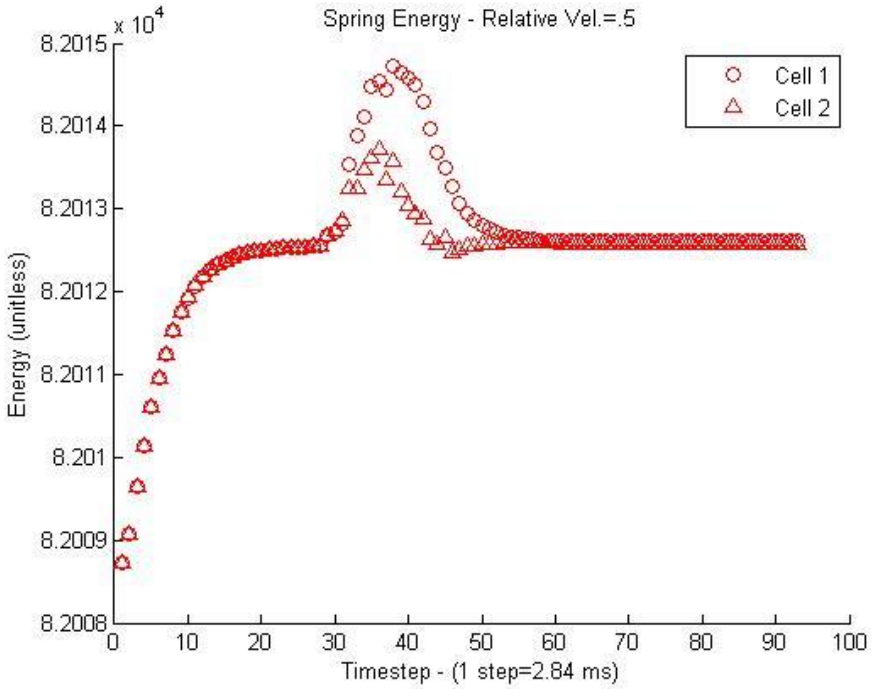

Figure 85: Spring potential energy of *K_s*=200 rotated collision simulation.

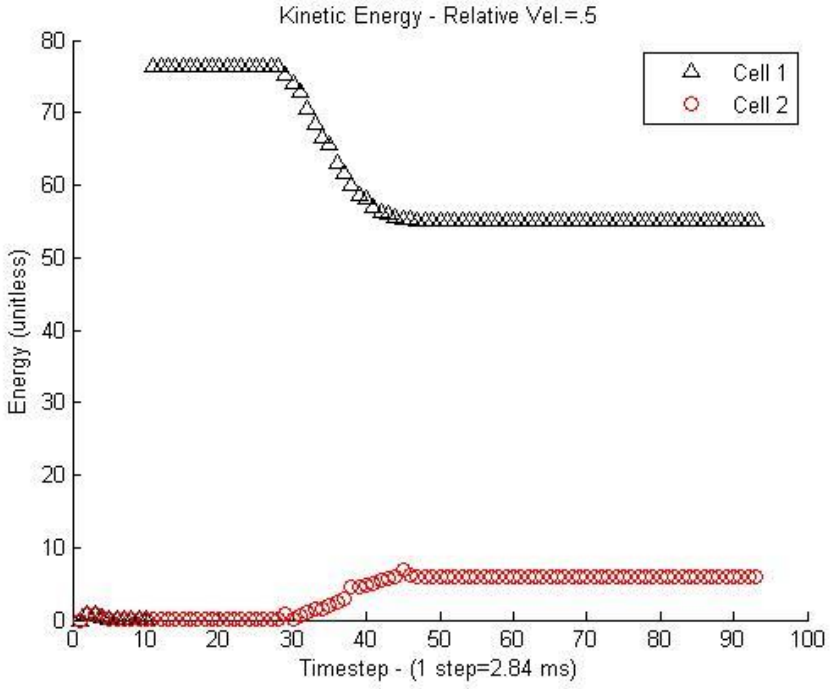

Figure 86: Kinetic energy of *K_s*=200 rotated collision simulation.

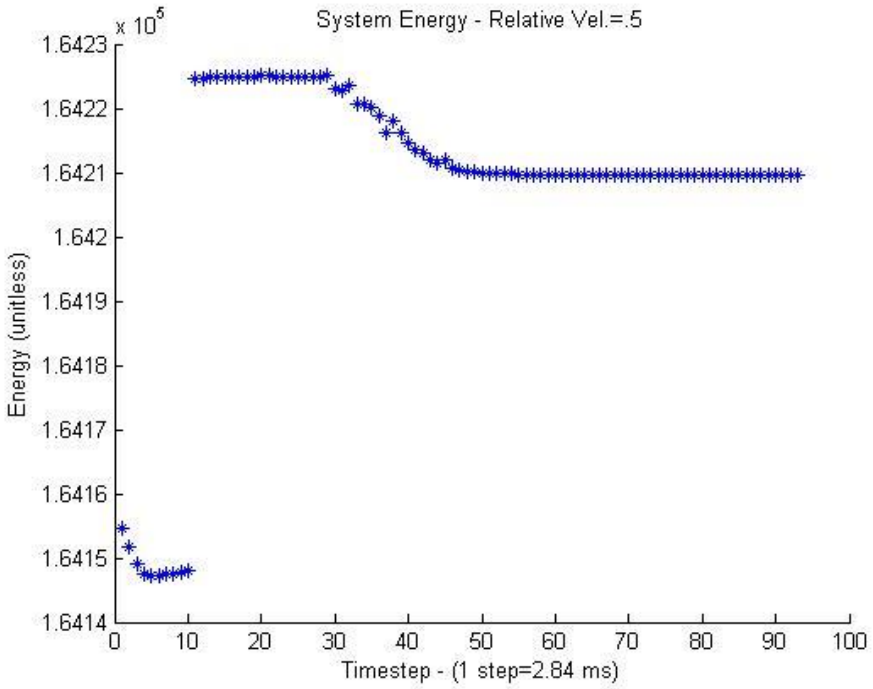

Figure 87: Total system energy of *K_s*=200 rotated collision simulation.

This simulation showcases the relative stability of this collision model over a large range of *K_s* values. As was the case with the direct collision simulations displayed earlier, this *K_s* value appears to be one of the last before the simulations become unstable and present unrealistic results. In this case, however, the spring potential energy terms remain stable as seen by their continuity, although some minor jumps can be seen in the potential energy curves, which led to the conclusion that we are now approaching the upper limit of *K_s*. The system energy is also interesting in that we can now see that the fluctuations caused by the collisions are actually large enough to be noticed in the area where energy is being lost to damping. The fluctuations are not large enough as to make the system energy appear discontinuous, so *K_s*=200 is still considered an acceptable value for use in simulations.

To be certain with our assumption that *K_s*=200 is the upper limit of acceptable *K_s* values, *K_s* was finally increased to 300 and the results can be seen below (Figure 88-Figure 91).

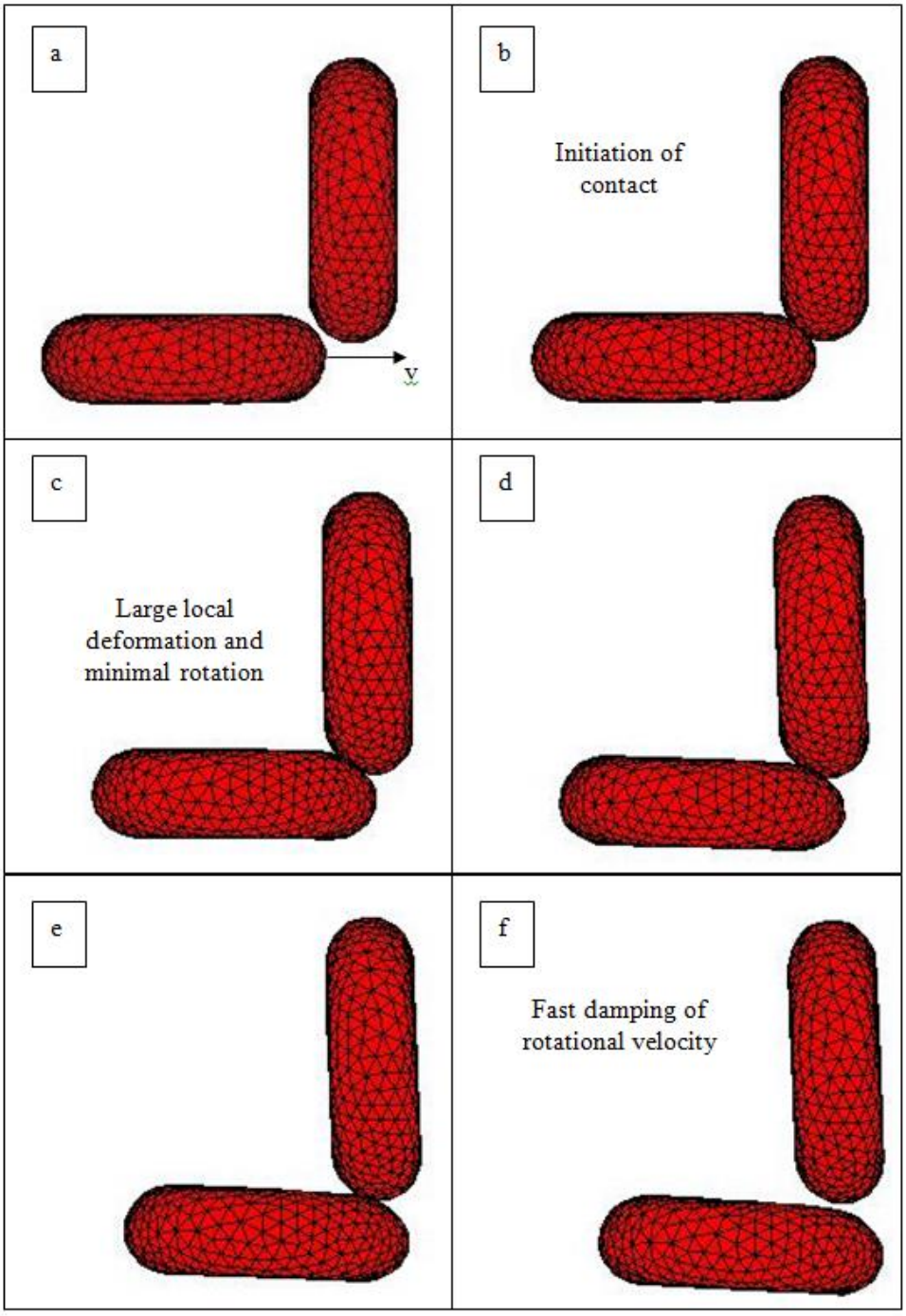

Figure 88: Shape evolution of the red blood cells in the rotated cell collision test case with *K_s*=300. The progression starts from the top left and moves to the right. Each progression corresponds to *Δt*=28.4ms.

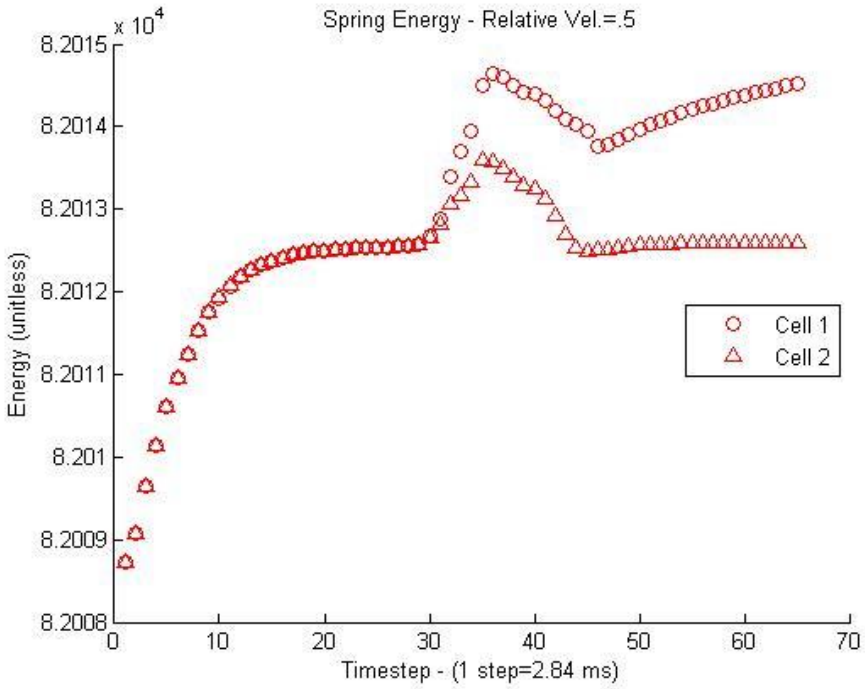

Figure 89: Spring potential energy of *K_s*=300 rotated collision simulation.

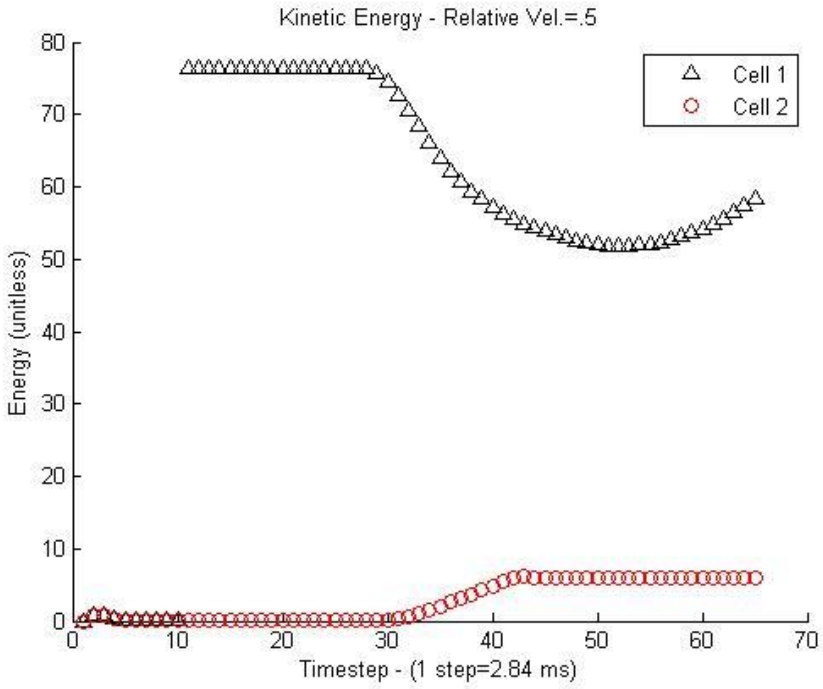

Figure 90: Kinetic energy of *K_s*=300 rotated collision simulation.

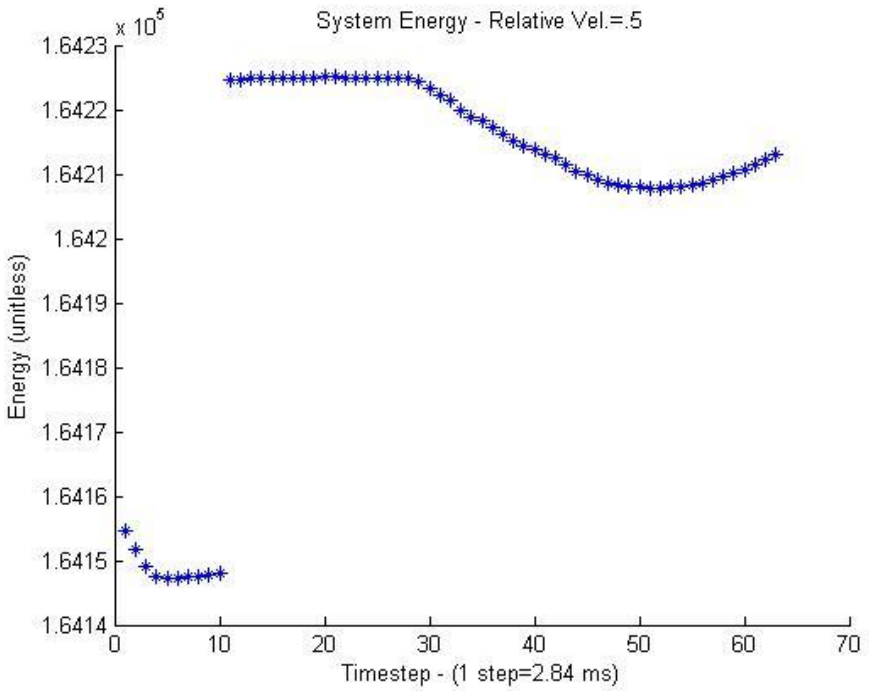

Figure 91: Total system energy of *K_s*=300 rotated collision simulation.

As was expected, increasing the *K_s* value to 300 introduces some abnormalities throughout the simulation. The first, and most notable, is that the cells, upon collision, begin rippling similarly to the rippling effect seen in Figure 65. This suggests that the force being applied to the nodes is too high. This can be clearly seen in the spring potential energy terms. The force is great enough that the spring energy potentials never actually settle to their pre-collision state. This is the type of instability that this simulation is attempting to identify and prevent in presenting feasible *K_s* ranges. Thus, the *K_s* value should not exceed 300 and should only be allowed to be marginally above 200. This range is consistent with that which is expected in the direct collision simulations which may help in reaching the conclusion that a single *K_s* value can handle collisions of two red blood cells at any angle.

Section 5.2.2: Rotational Effects

The angular velocity of the first red blood cell is plotted below (Figure 92-Figure 95).

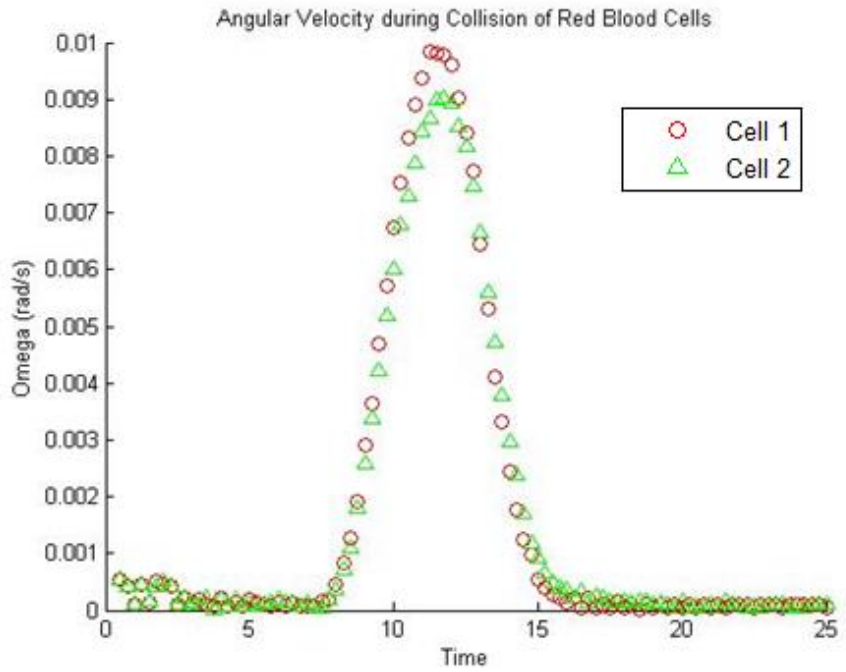

Figure 92: Rotational velocity of red blood cells during rotated collision simulation with K_s=10. Each time step is Δt=2.84ms.

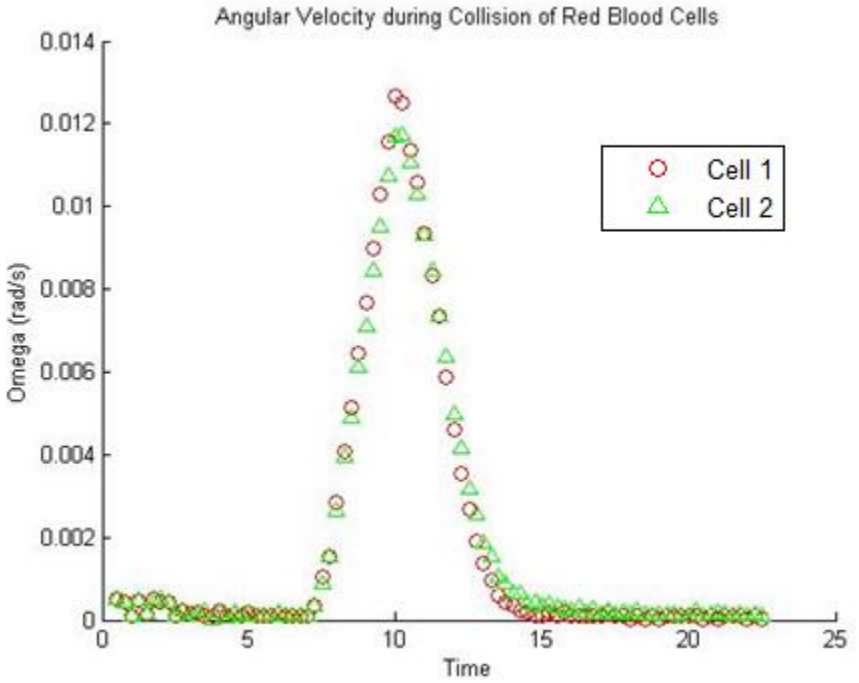

Figure 93: Rotational velocity of red blood cells during rotated collision simulation with K_s=75. Each time step is Δt=2.84ms.

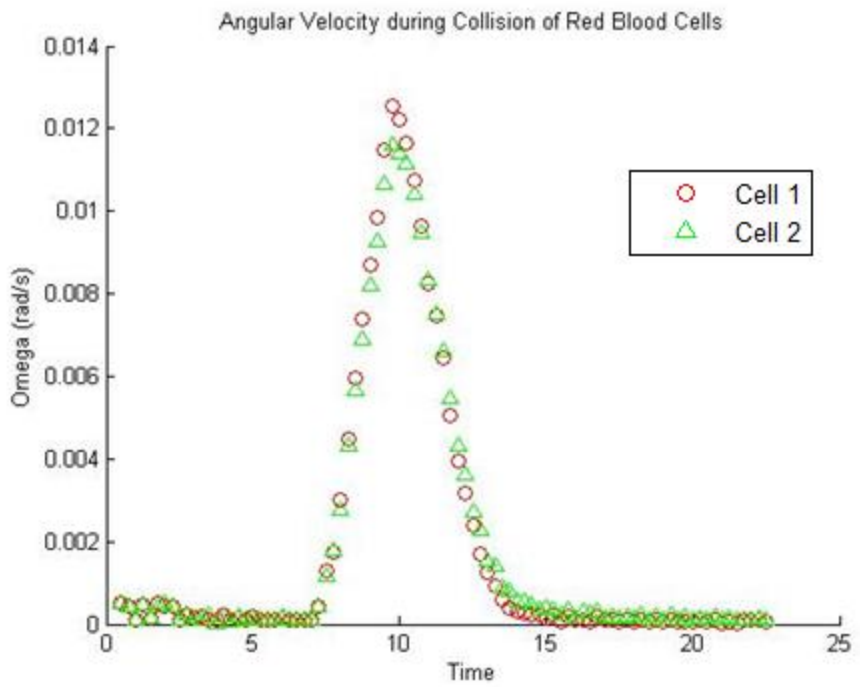

Figure 94: Rotational velocity of red blood cells during rotated collision simulation with K_s=100. Each time step is Δt=2.84ms.

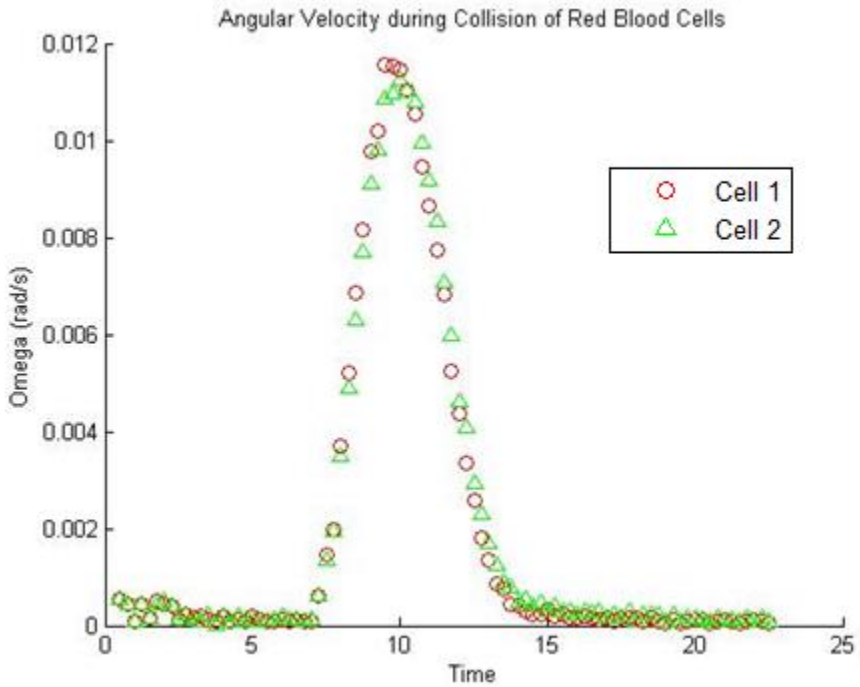

Figure 95: Rotational velocity of red blood cells during rotated collision simulation with K_s=200. Each time step is Δt=2.84ms.

Examining the rotational velocity of these simulations reveals certain phenomena that are of interest. The first noteworthy observation comes from the magnitude of the

maximum angular velocity of the cells between each simulation. As would be expected, the maximum angular velocity is higher in *K_s*=75 than in *K_s*=10. Because *K_s* directly corresponds to the force applied to the collided nodes, there should be a higher moment on the cell which creates a higher angular velocity. Also, because *K_s*=100 is reasonably close to *K_s*=75 in terms of collision force, the total collision force is on a similar scale, and the similarities in rotational velocity support that. However, in examining the *K_s*=200 rotational velocity, we see that the maximum rotational velocity is actually lower than for the *K_s*=75 and 100 simulations. One can formulate two theories that are believed to influence the rotational velocity dropping as *K_s* increases. The first is that the larger *K_s* prevents larger collision volumes which results in fewer nodes colliding between two cells. The collision force applied on the fewer nodes may be on the same scale as the simulations with smaller *K_s* values, but because the force is concentrated on fewer nodes, the membrane damping more effectively prevents the propagation of collision force from the collided nodes throughout the cell. The more concentrated force, therefore, is unable to affect the overall cell more than the more dispersed force of similar magnitude. The second theory is an extension of the rippling phenomena that was observed in the *K_s*=300 simulations. The *K_s*=200 simulations did not show overt signs of this rippling effect, though it is still possible that the force was already excessive which resulted in unphysical ripples which would again lead to unrealistic damping. The exaggerated damping force would lead to a smaller moment for rotation than the simulations with smaller K_s values.

Second, it is worth noting that the rotational velocity approaches zero after a short period of time. This is due to the type of membrane damping that is implemented. The membrane damping in this model is directly related to the relative velocity between two nodes as outlined in Section 1.3. In rotational motion, relative velocity is necessarily non-zero, even if link length is constant. This is clear because if we examine rotational motion when the center of mass is fixed, the following equation becomes valid:

$$\overline{v_i} = \overline{\omega} \times \overline{r_i} \tag{5.3}$$

Because ω is constant with the cell rotating around its center of mass, the r_i term will change for each point on the blood cell's surface. With this, every velocity on the red blood cell surface will be different, so the relative velocities between neighboring nodes will also be non-zero. This results in a damping force which effectively removes rotational effects from the simulation as is seen in the results above.

Section 5.3: Sedimentation Collision

Red blood cell sedimentation is of interest to researchers because red blood cell sedimentation rate can be used to diagnose certain diseases. For this reason, the collision method was tested to see how well it could handle sedimentation collisions. It was originally believed that the biconcave shape would cause problems in calculating volume which would, in turn, cause unrealistic dynamics. It was anticipated that the biconcave shape would lead to a large empty volume between the two colliding cells that would give an over-approximation of the collision volume due to the ellipsoid approximation (The ellipsoid would encompass a large amount of the volume that is actually just empty space as in Figure 96).

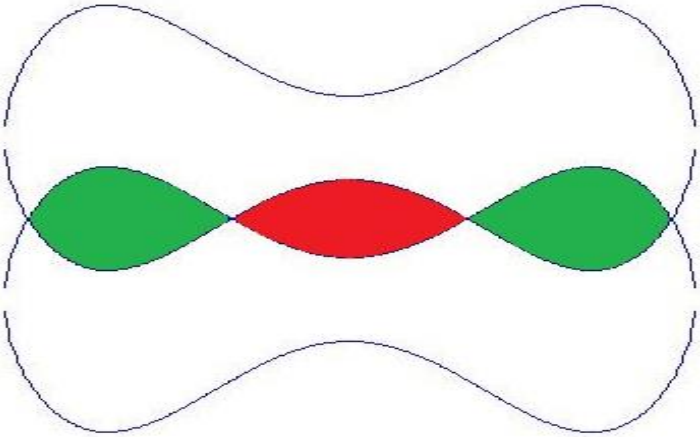

Figure 96: 2-D representation of the sedimentation contact. The green area represents area that should contribute to collision force. However, due to the ellipsoid approximation, some of the red area is considered. The red area does not have intersected blood cell area.

Therefore, there was legitimate concern that the method's weakness would come from one of the simulations that is of the greatest interest to researchers. To alleviate this concern, a simple test setup was devised in which one cell "falls" directly on top of another, representing the worst case scenario in a sedimentation simulation shown in Figure 97.

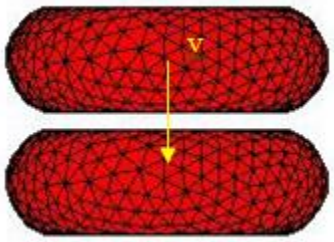

Figure 97: Initial test setup of the red blood cell sedimentation simulations.

As was the case with the previous two simulations, conservation of energy and continuity of the spring potential energy are of primary concern to the simulations.

Section 5.3.1: Energy Conservation and Dissipation

The sedimentation collision results for *K_s*=10 are shown below in Figure 98-Figure 101.

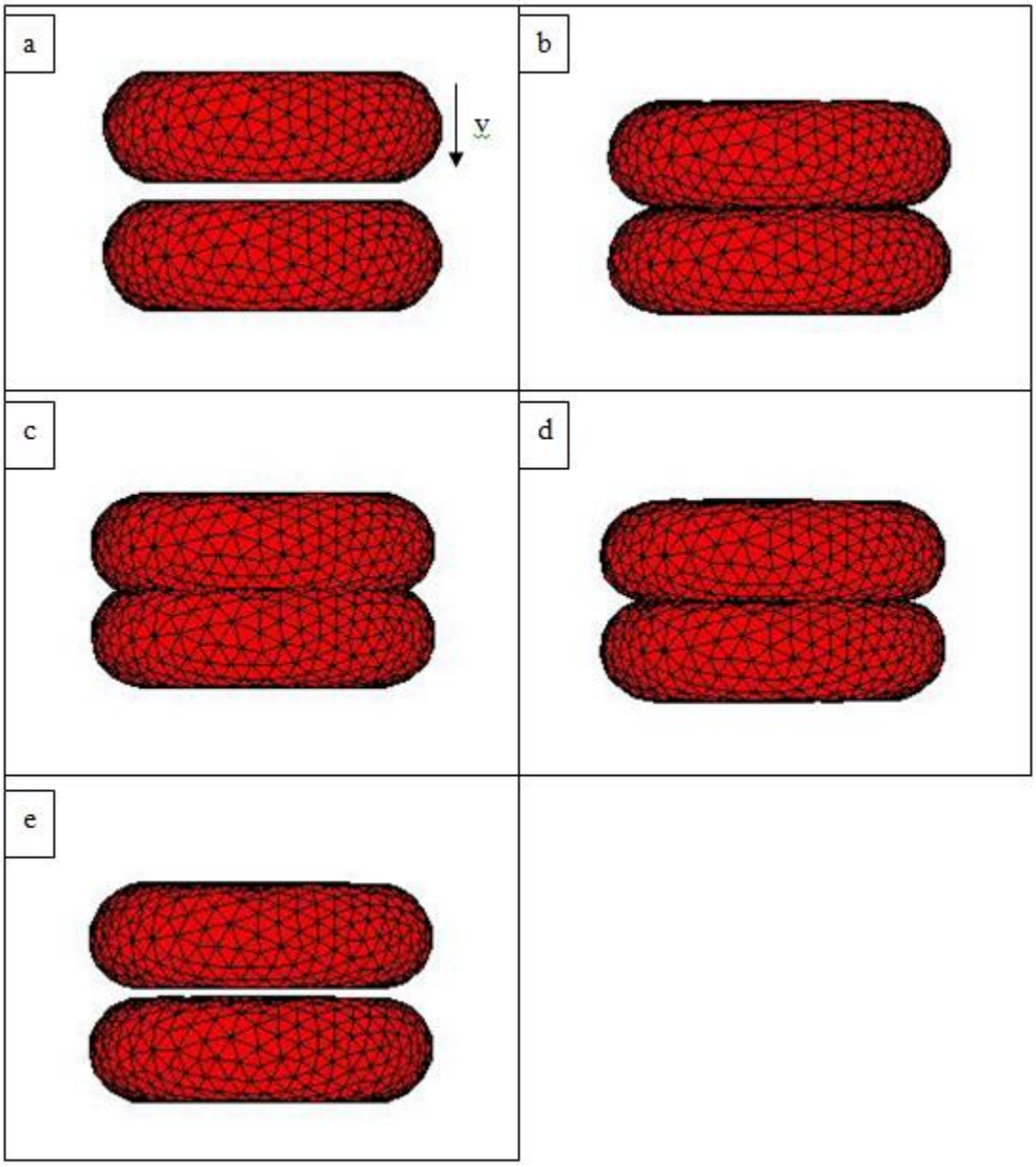

Figure 98: Shape evolution of the red blood cells in the sedimentation test case with *K_s*=10. The progression starts from the top left and moves to the right, as numbered. Each progression corresponds to *Δt*=28.4ms.

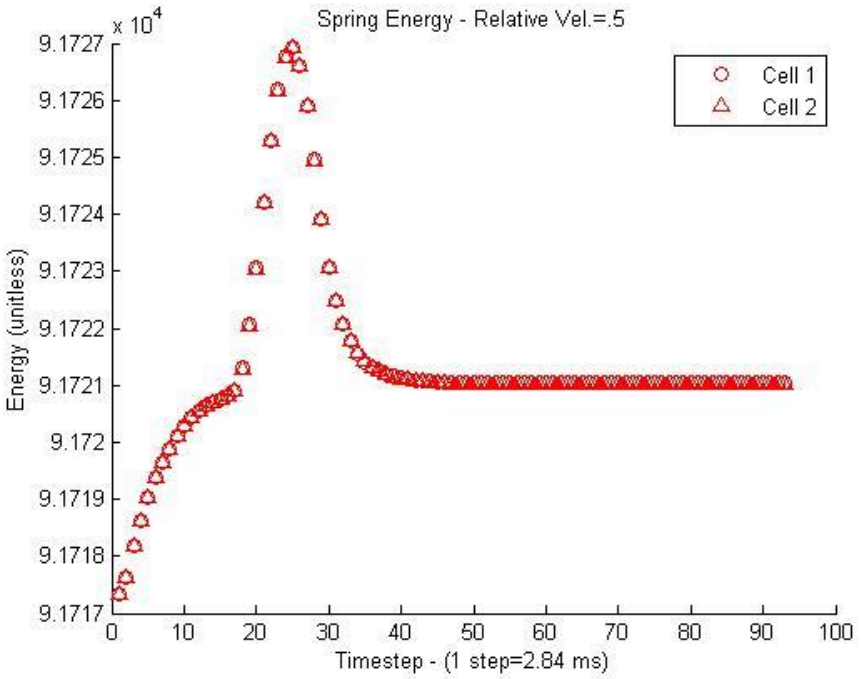

Figure 99: Spring potential energy of *K_s*=10 sedimentation simulation.

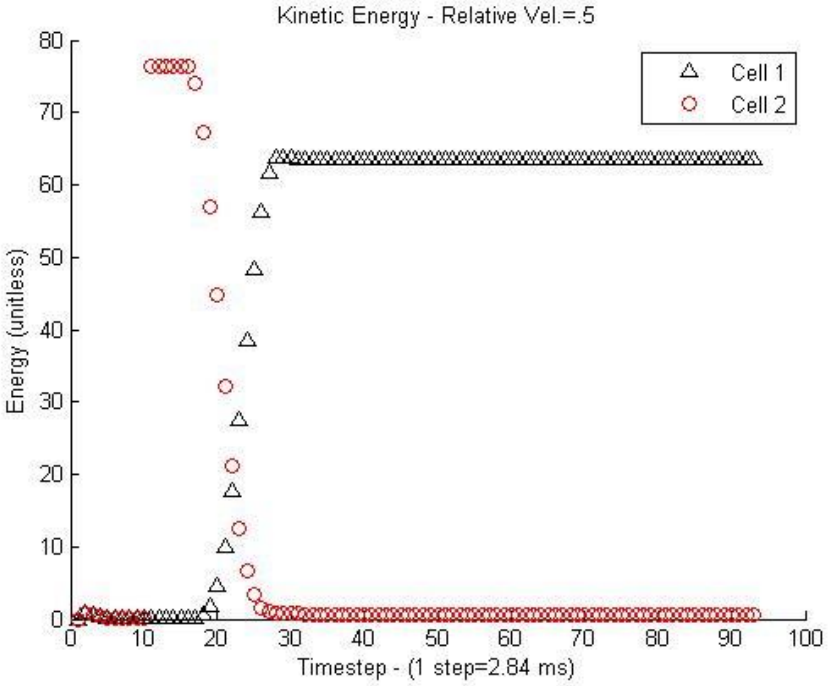

Figure 100: Kinetic energy of *K_s*=10 sedimentation simulation.

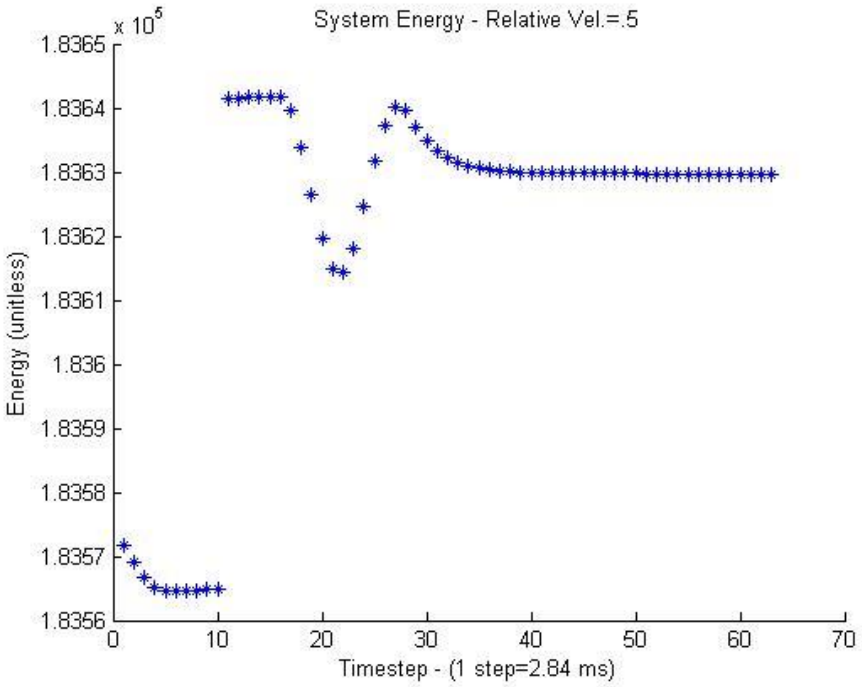

Figure 101: System energy of *K_s*=10 sedimentation simulation.

It is immediately clear that the spring potential energy terms in the *K_s*=10 simulation are continuous. This would suggest that the simulations are stable under the given conditions. It should be pointed out that the spring potential energy terms in this simulation are much steeper during collision than in the previous two simulations. This has to do with the large jump in collision volume upon contact. While the first two simulations dealt with relatively small contact volumes because the contact surface was small, the resting contact simulation has a significantly larger contact surface, and with that, there is a larger contact volume. It is, however, encouraging to know that the collision is stable despite this large jump in volume.

In addition, the system energy decreases as a result of the collision which would indicate that the energy of the system is conserved with the loss coming from the membrane damping terms. It has thus been established that the red blood cell

collision model is effective in dealing with sedimentation, but it should be confirmed that the range of *K_s* values is similar to that of the previous simulations. If the same range cannot be confirmed, then a new total range will be proposed. *K_s*=75 was the first simulation run and the data is presented in Figure 102-Figure 105.

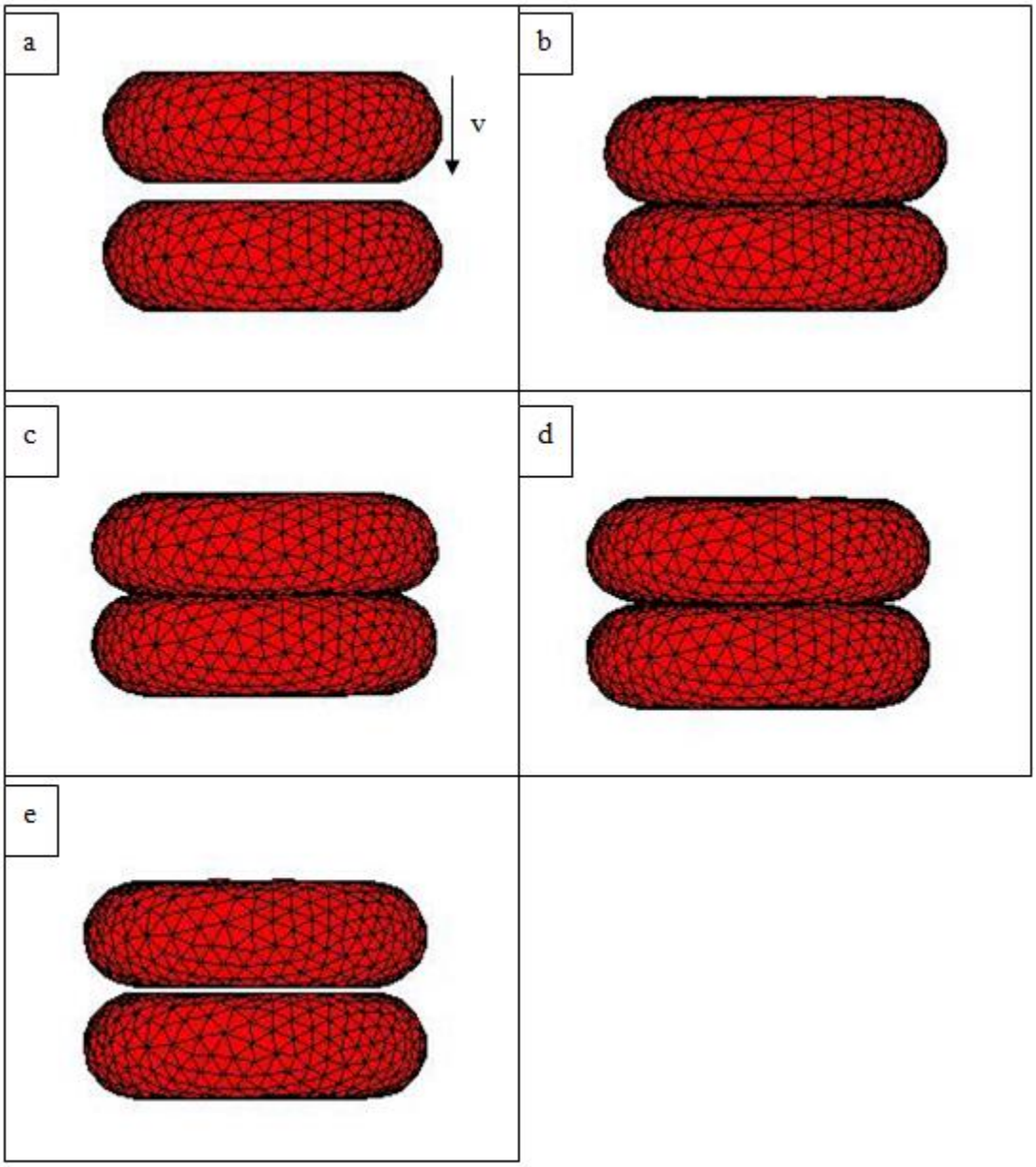

Figure 102: Shape evolution of the red blood cells in the sedimentation test case with *K_s*=75. The progression starts from the top left and moves to the right, as numbered. Each progression corresponds to *Δt*=28.4ms.

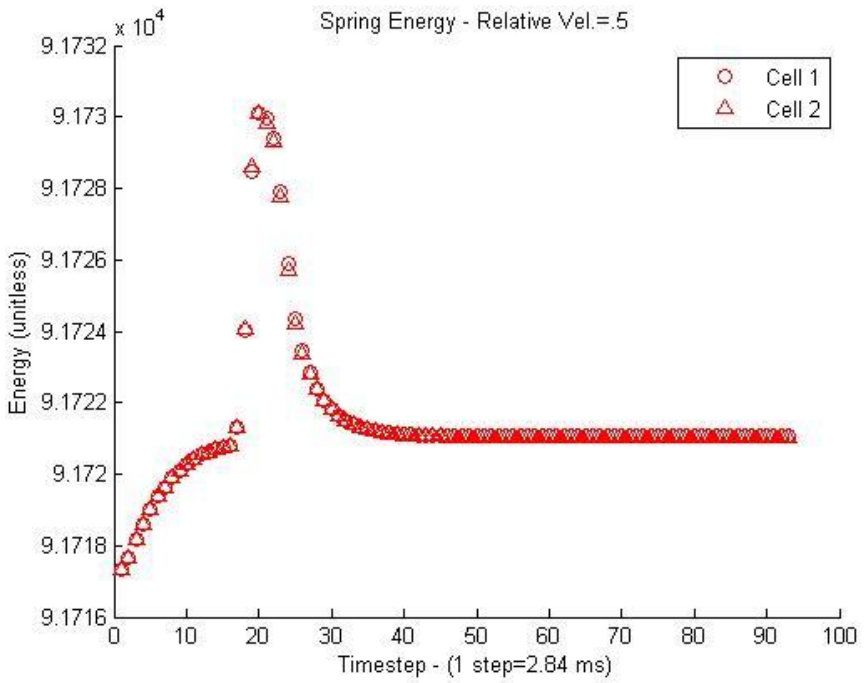

Figure 103: Spring potential energy of *K_s*=75 sedimentation simulation.

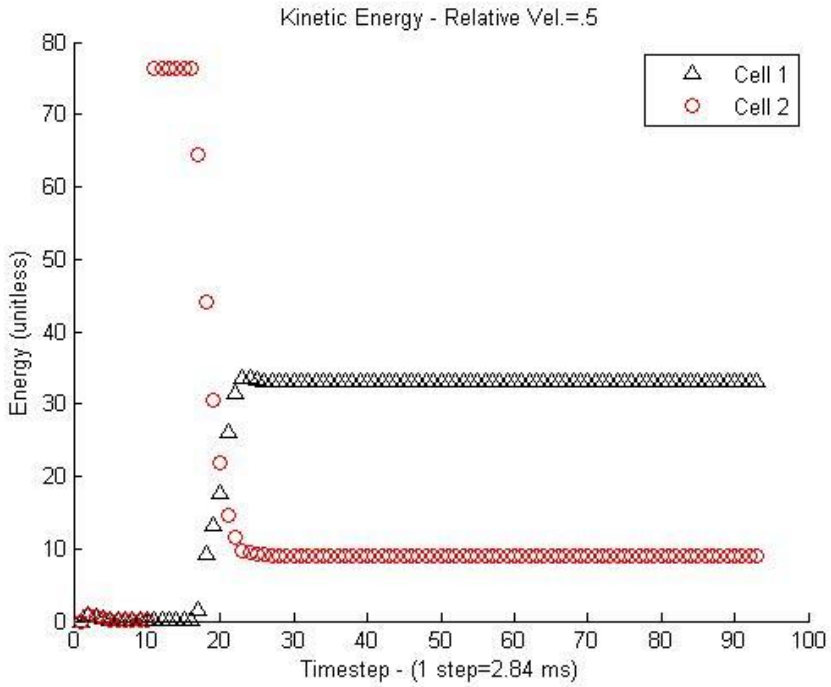

Figure 104: Kinetic energy of *K_s*=75 sedimentation simulation.

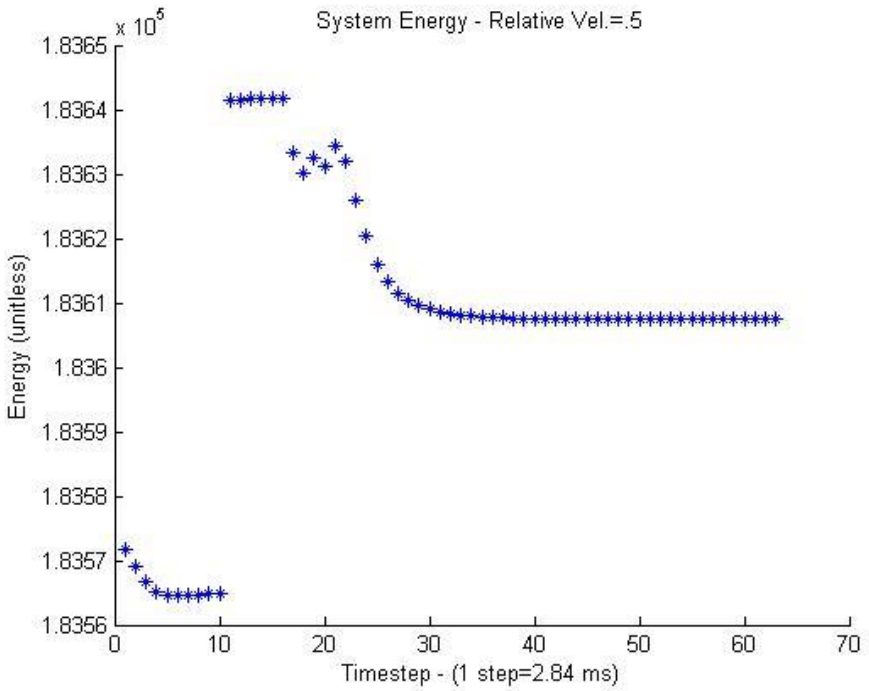

Figure 105: Total system energy of *K_s*=75 sedimentation simulation.

In using *K_s*=75 as an intermediate step that represents the value that gives the appearance of most realistic physical results, the simulation is again stable. It also appears that the physical results seem feasible with respect to the amount of deformation seen during the collision.

When examining the spring potential energy terms, the terms are slightly more volatile than the *K_s*=10 simulation but none of these fluctuations are so large that the smoothness of the curve would suggest discontinuity. Therefore, it is reasonable to conclude that the stability of this simulation is acceptable at this time scale under the given conditions.

Likewise, to be able to conclude that the simulation is physically feasible, the energy of the system should be examined to check that energy is conserved. As was the case

with the previous simulation, the potential energy terms return to equilibrium values while the kinetic energy significantly drops due to the collision, so the total energy is conserved due to the energy dissipation. The *K_s*=100 simulation results are presented below in Figure 106-Figure 109.

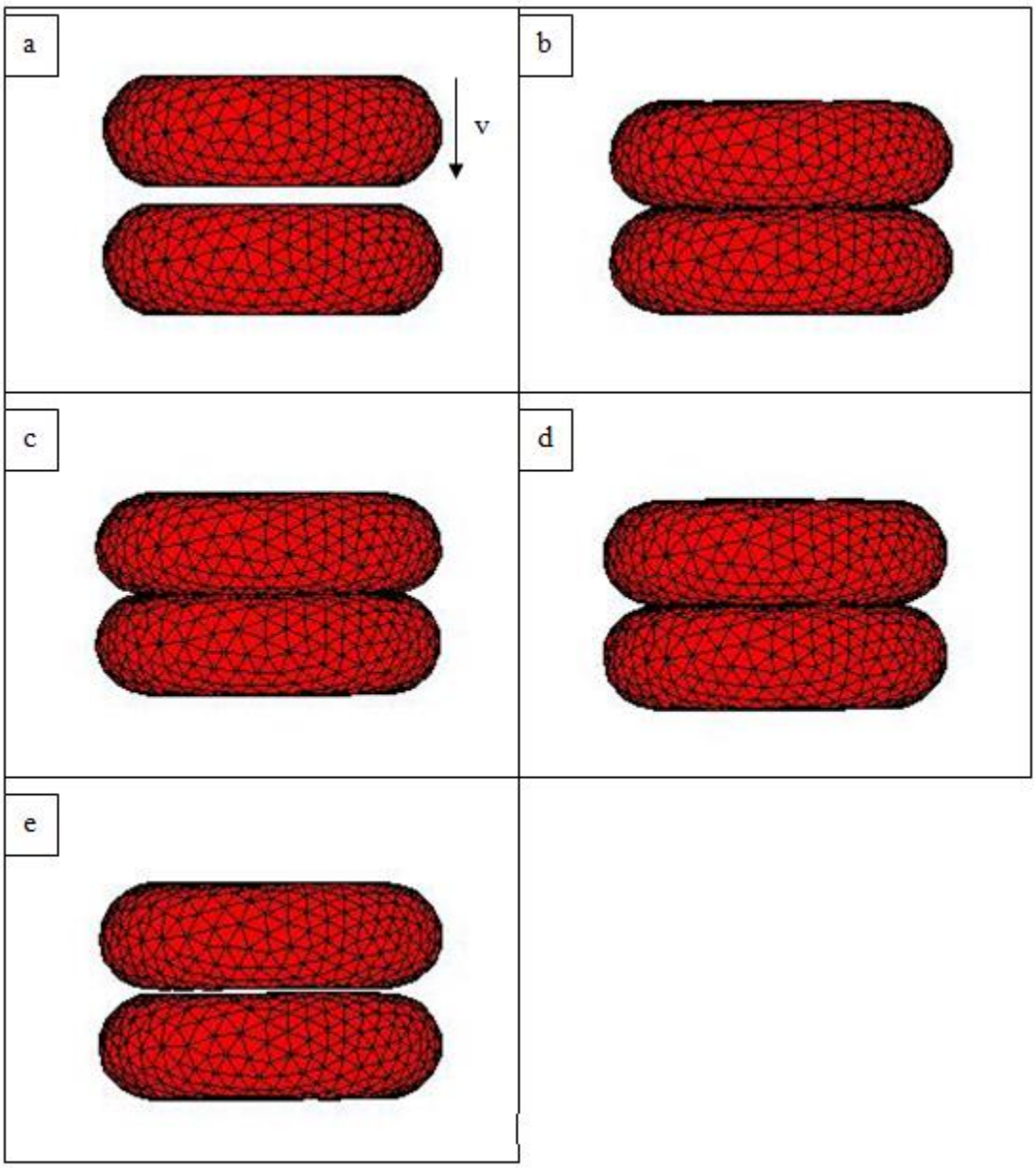

Figure 106: Shape evolution of the red blood cells in the sedimentation test case with *K_s*=100. The progression starts from the top left and moves to the right, as numbered. Each progression corresponds to *Δt*=28.4ms.

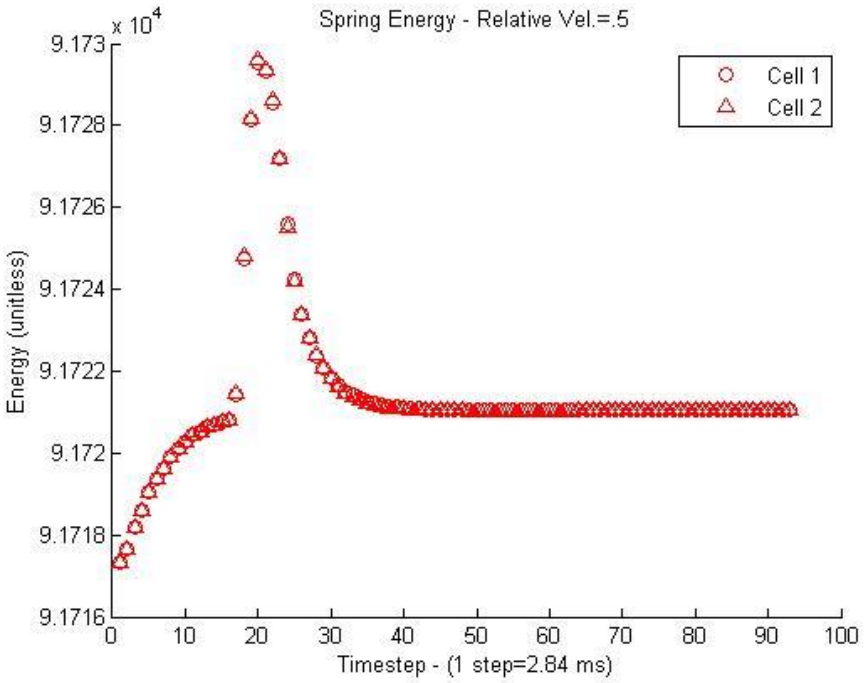

Figure 107: Spring potential energy of *K_s*=100 sedimentation simulation.

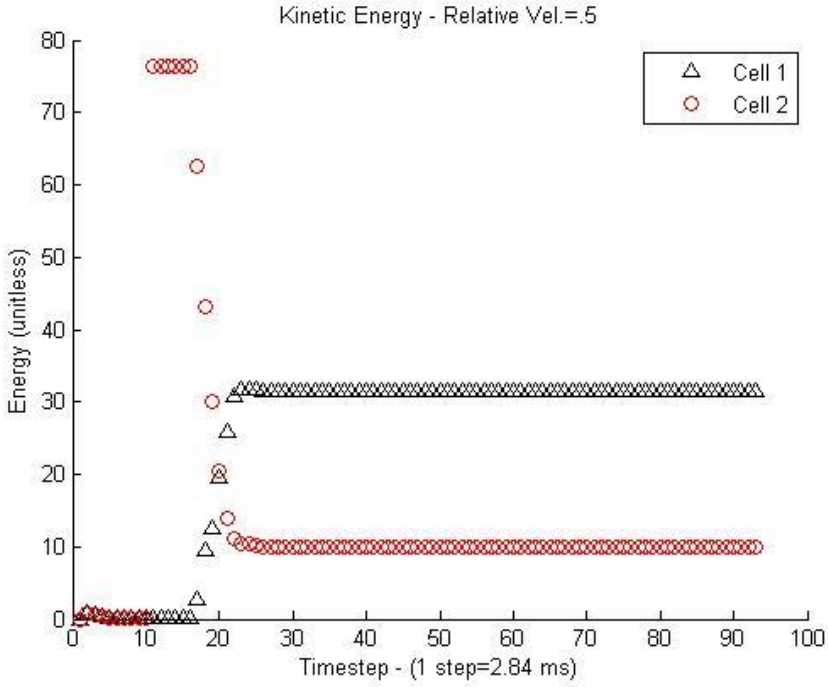

Figure 108: Kinetic energy of *K_s*=100 sedimentation simulation.

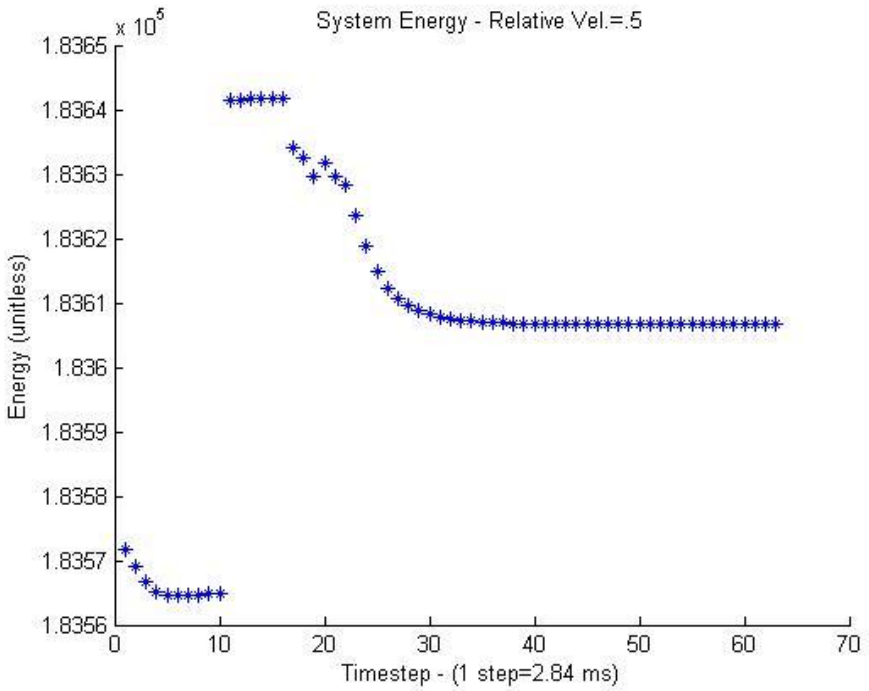

Figure 109: Total system energy of *K_s*=100 sedimentation simulation.

As would be expected, the *K_s*=100 simulation closely resembles the *K_s*=75 simulation, so for brevity, the analysis of these simulations will be relatively short. The spring potential energy terms are smooth which would suggest stability in the simulation while the energy lost due to membrane damping in the kinetic energy is greater than the fluctuation of energy in the potential energy terms. This suggests that the simulations are acceptable for sedimentation with *K_s*=100. The results in Figure 110-Figure 113 correspond to simulations with *K_s*=200.

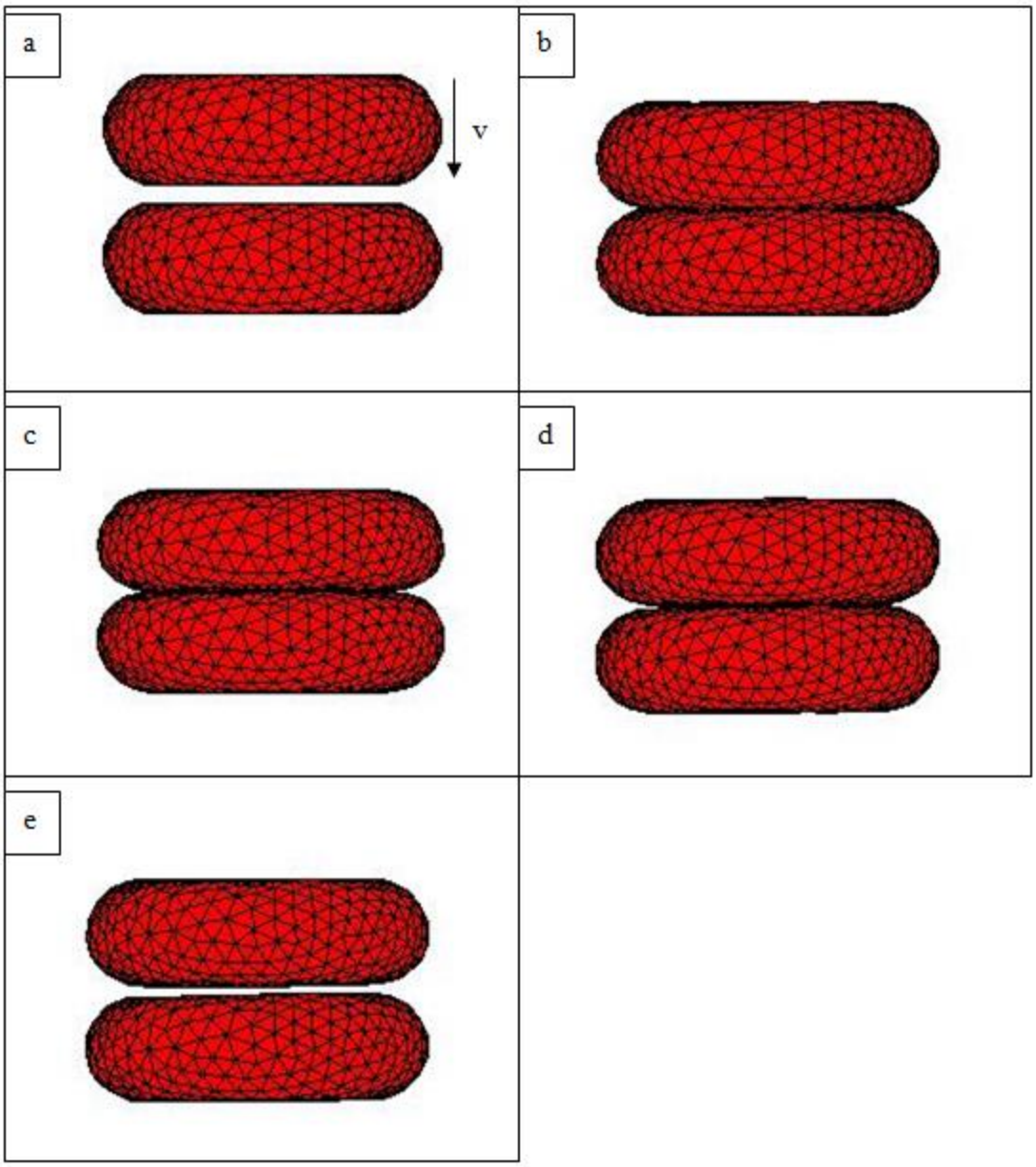

Figure 110: Shape evolution of the red blood cells in the sedimentation test case with *K_s*=200. The progression starts from the top left and moves to the right, as numbered. Each progression corresponds to *Δt*=28.4ms.

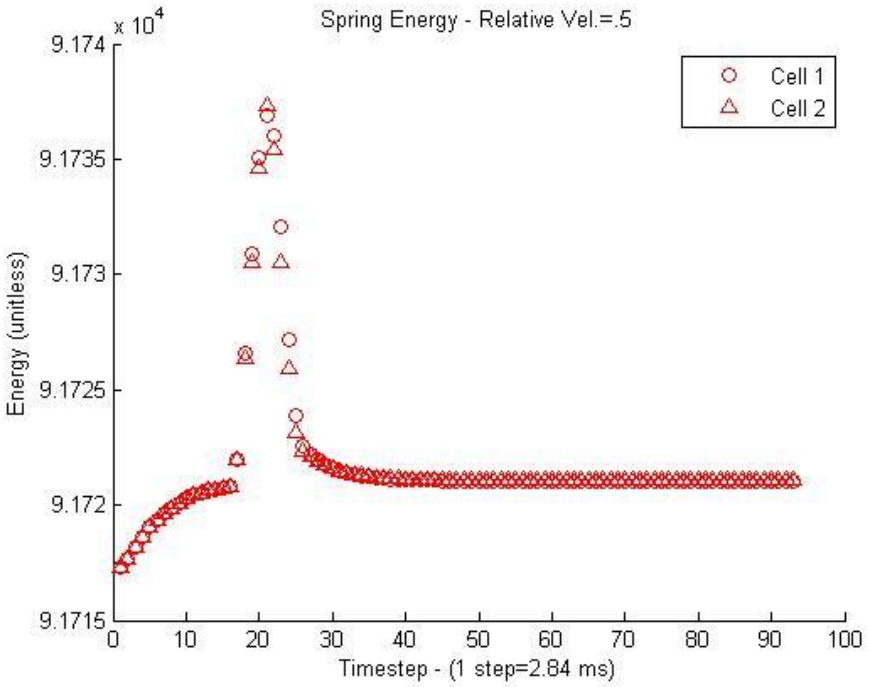

Figure 111: Spring potential energy of *K_s*=200 sedimentation simulation.

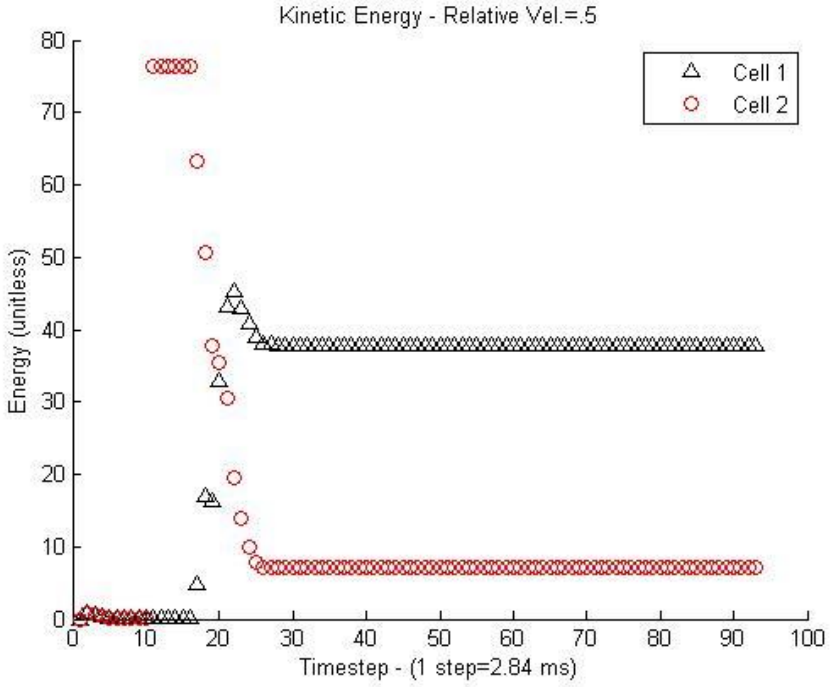

Figure 112: Kinetic energy of *K_s*=200 sedimentation simulation.

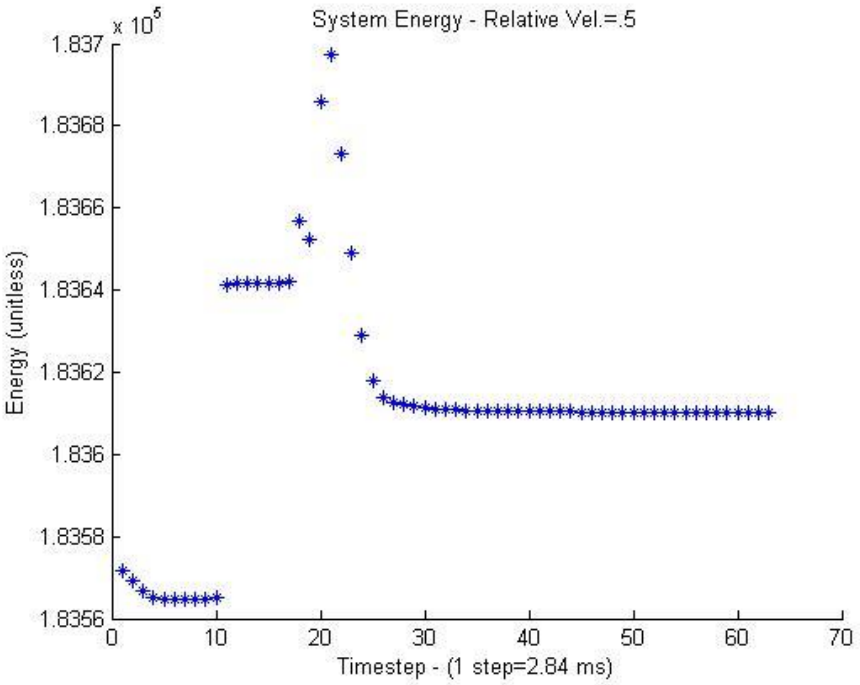

Figure 113: Total system energy of *K_s*=200 sedimentation simulation.

It was at the *K_s*=200 value that the hypothesis for instability, suggested at the beginning of this section, was expected to materialize. It can be seen, however, that the large jump in intersect volume does not lead to instability at *K_s*=200. Though the change in the spring potential energy terms is sharper than previous simulations, there are no jumps that would suggest discontinuities or instability. The smoothness and continuity of these potential energy terms suggests that the sedimentation simulations may not require a different range of *K_s* values as was originally hypothesized.

Additionally, the amount of energy lost due to membrane damping is greater than the change of energy due to the collision. The conservation of energy is satisfied with the *K_s*=200 simulations, so it is a viable option for not only sedimentation, but for all

types of collisions a red blood cell may experience. As a final step, the *K_s*=300 was checked for viability in this type of collision (see Figure 114-Figure 117).

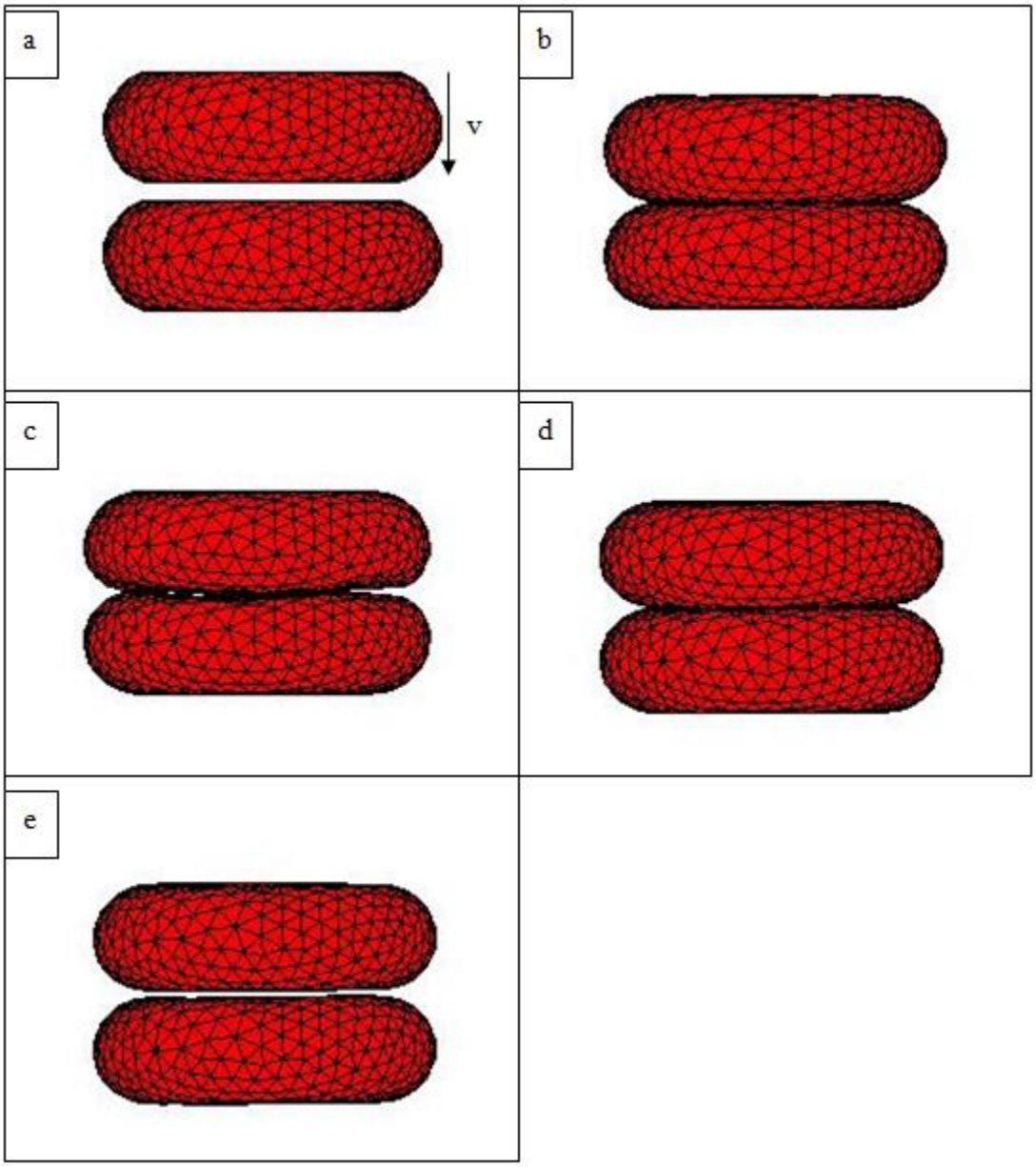

Figure 114: Shape evolution of the red blood cells in the sedimentation test case with *K_s*=300. The progression starts from the top left and moves to the right, as numbered. Each progression corresponds to *Δt*=28.4ms.

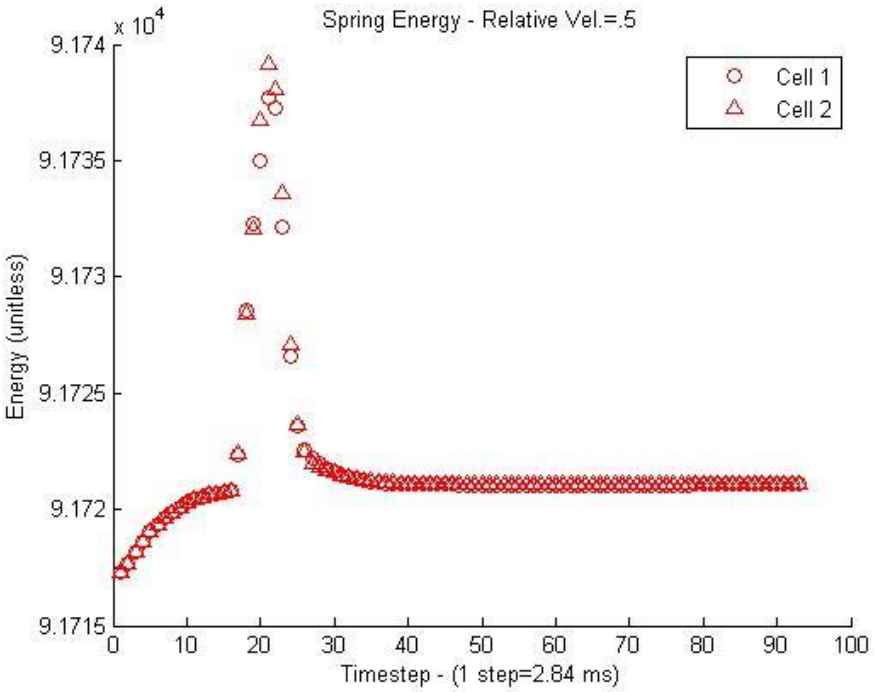

Figure 115: Spring potential energy of *K_s*=300 sedimentation simulation.

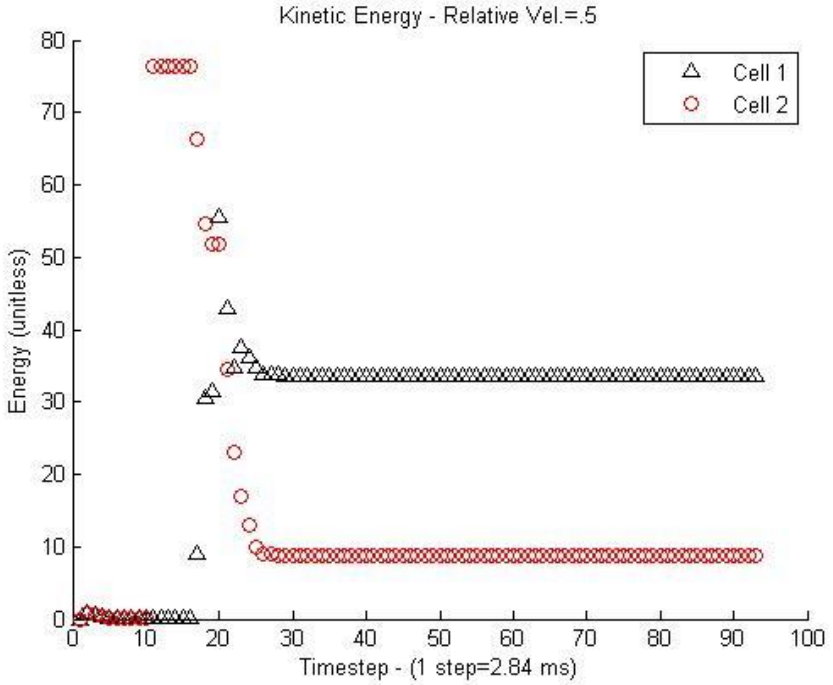

Figure 116: Kinetic energy of *K_s*=300 sedimentation simulation.

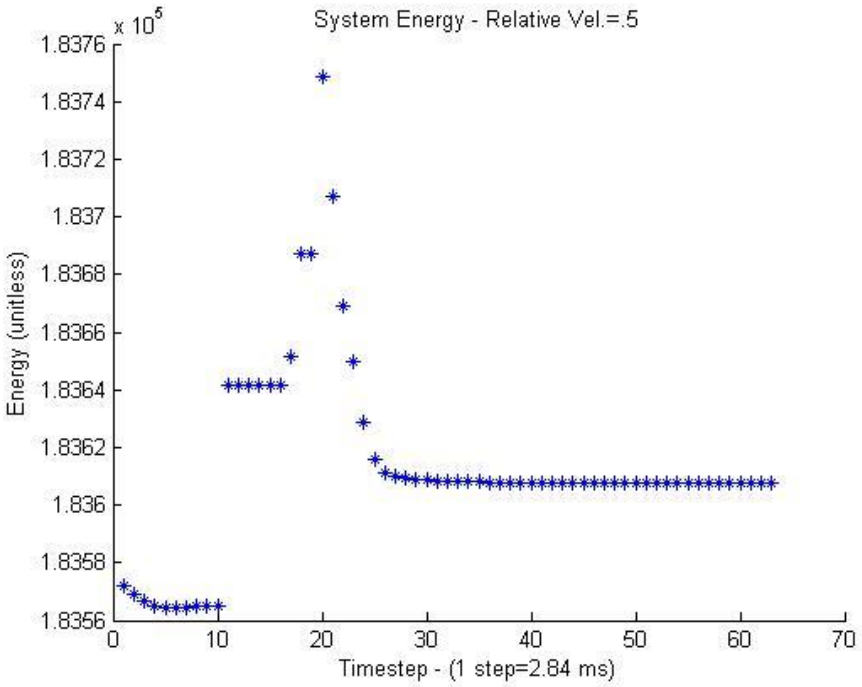

Figure 117: Total system energy of *K_s*=300 sedimentation simulation.

Though the spring potential energy terms in this simulation appear to be smooth and continuous, the problem with this simulation can be found in Figure 114. A clear gap can be seen in the third image between the two cells that remain in contact through the next image. As was the case with previous *K_s*=300 simulations, the gap is a result of the rippling effect from such a large force due to the high *K_s* constant. The rippling effect is clearly not physically feasible, so disregarding *K_s*=300 for the sedimentation contact is consistent with the other simulation presented earlier.

Section 5.3.2: Effective Coefficient of Restitution

As was the case with the direct collision simulations, it is helpful to analyze the effective coefficient of restitution of the system with the defined parameters. The analysis of the coefficient of restitution at a different angle may present a different range of coefficients of restitution which would imply that the effective coefficient of

restitution is dependent on the collision geometry. The following images (Figure 118-Figure 121) track the velocities centers of mass throughout the simulations which are regarded as the total velocities of the red blood cells.

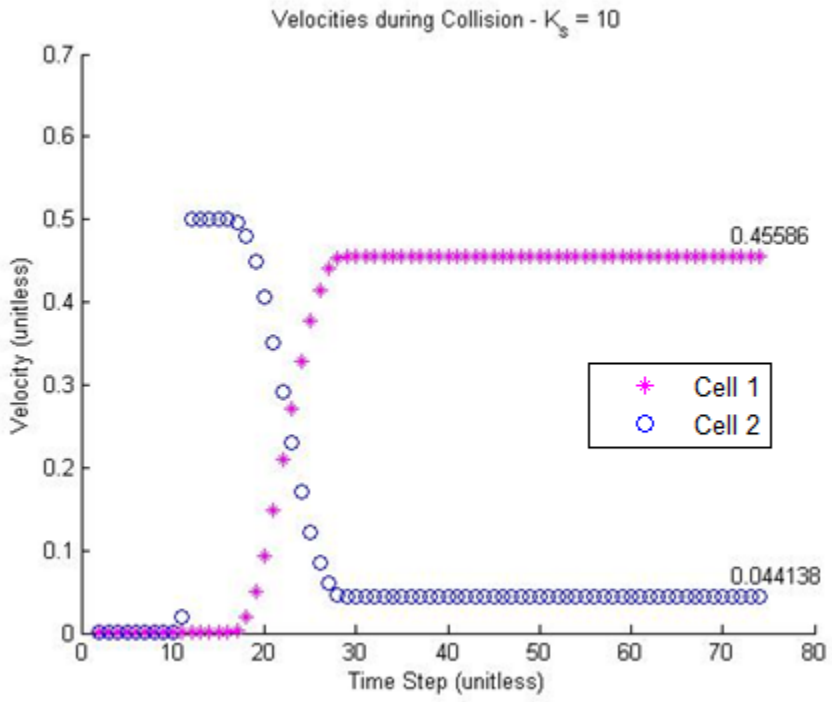

Figure 118: Velocity of the red blood cells throughout the *K_s*=10 sedimentation simulation.

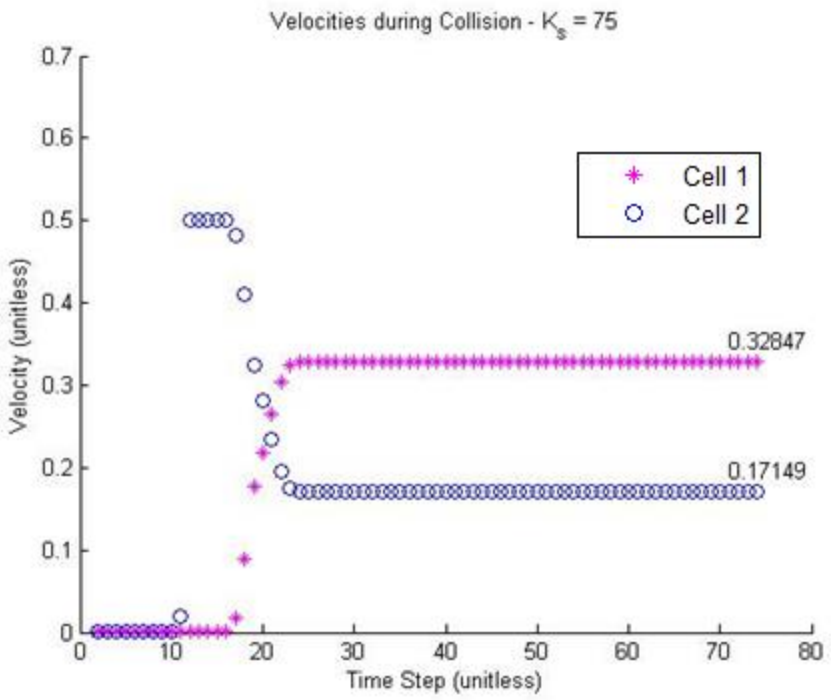

Figure 119: Velocity of the red blood cells throughout the *K_s*=75 sedimentation simulation.

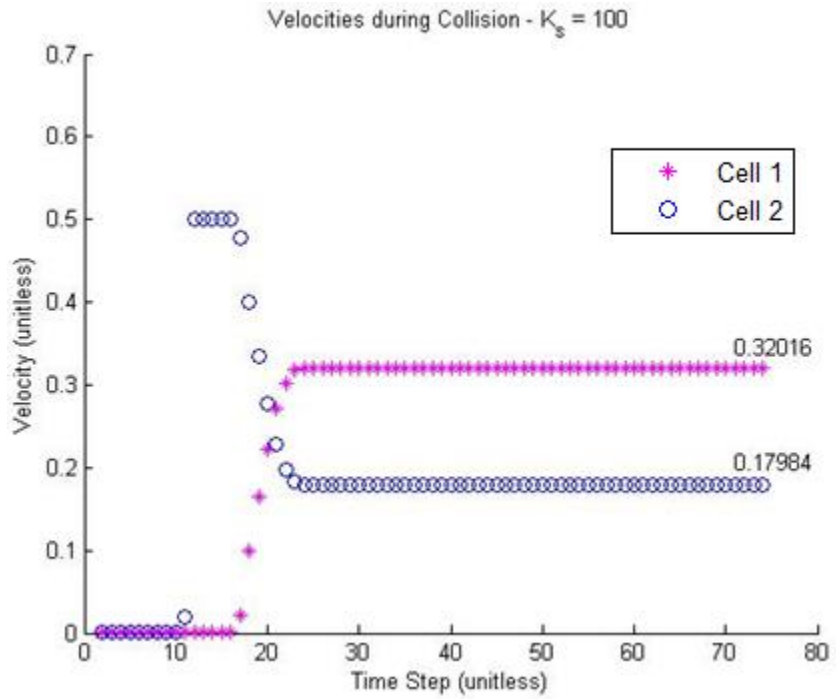

Figure 120: Velocity of the red blood cells throughout the *K_s*=100 sedimentation simulation.

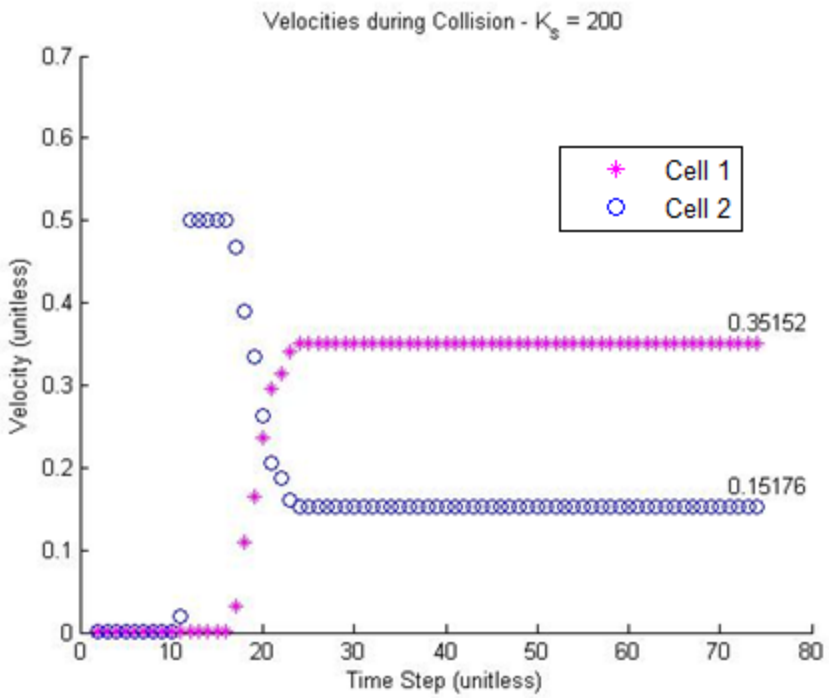

Figure 121: Velocity of the red blood cells throughout the *K_s*=200 sedimentation simulation.

The values printed on the far right side of each image correspond to the final velocity of each red blood cell which will be used in the calculations that follow. From the velocities of the direct collision, the effective coefficient of restitution can be calculated as it was done in Section 5.1.1. Because the collisions result in a negligible amount of rotation, it is a fair assumption to treat these in a similar manner as rigid spheres colliding would be treated to find a coefficient of restitution. In calculating these values, the table below can be created.

K_s	*ECOR*
10	.8234
75	.3140
100	.2807
200	.3994

Table 3: Observed values of the effective coefficient of restitution in sedimentation contact.

It can be seen that the *ECOR* values that are reported above differ from those shown in the direct collision results. This is reasonable because the larger contact area results in larger repulsive forces that prevent the nodes from remaining in contact as long as the direct collision simulations which allows for shorter times to propagate the force through the cell which results in the total velocity of the cell. Due to the unique shape and semi-rigid behavior of the blood cell acting as a fluid-filled sac with a rigid spectrin network, it is recommended that experiments be carried out to determine how different orientations of collision may result in different transmission of velocities between red blood cells.

Another important note to acknowledge is that, again, the *ECOR* appears to rise at *K_s*=200 after rapidly dropping for the previous values. However, similarly to the direct collision observation, the conservation of momentum is violated with the *K_s*=200 simulation as the final momenta are greater than the initial momentum of the system. The conservation of momentum is violated, so the simulation should not be considered valid. However, because the range of *ECOR*s is large, further simulations were not deemed necessary.

Section 5.3.3: Sedimentation Analysis and Resting Contact

Red blood cell sedimentation is a phenomenon in which red blood cells settle when placed in a narrow tube [47, 48]. The sedimentation rate can be used to determine the presence of irregularly shaped red blood cells that lead to higher adhesion rates between cells. The higher adhesion leads to higher density areas of blood cells, which in turn leads to faster settling rate. Because sedimentation rate is important in

the diagnosis of certain diseases, understanding the mechanisms of the red blood cell sedimentation is an area of great interest to researchers. In creating a collision model, the groundwork has been created that will allow for full-scale simulations of blood cells during sedimentation. The rate at which cells settle is very slow, which implies that the contact model presented above would be more than capable of handling the magnitudes of relative velocity encountered in this type of simulation. It is important to understand that while the contact model proposed herein is able to handle contact between red blood cells, it was not specifically developed to handle resting contact, when two cells should stick together after contact.

Resting contact has two specific implementations when attempting to recreate blood cell physics. The first is the scenario represented above in which gravity is holding down a particle and another particle falls onto the first and gravity is strong enough to hold the two in contact. This type of resting contact requires that the force due to gravity is equal to the restoring force, not only due to the collision, but also due to the propagation of force throughout the discretized cell from the collision. It is for this reason that developing an equilibrium condition for this model is not as straightforward as the equilibrium imposed in a system such as [37], where force can be simply derived based on nodal penetration. From the aforementioned reference, Frenning derives the contact force that exactly cancels the penetration that the two bodies would typically have incurred. Special conditions can be placed in the code that was developed in this thesis to allow for constant contact that would approximate this equilibrium.

The second specific implementation of this resting contact pertains to rouleaux formation. Rouleaux are stacks of red blood cells that are found in a typical blood flow.

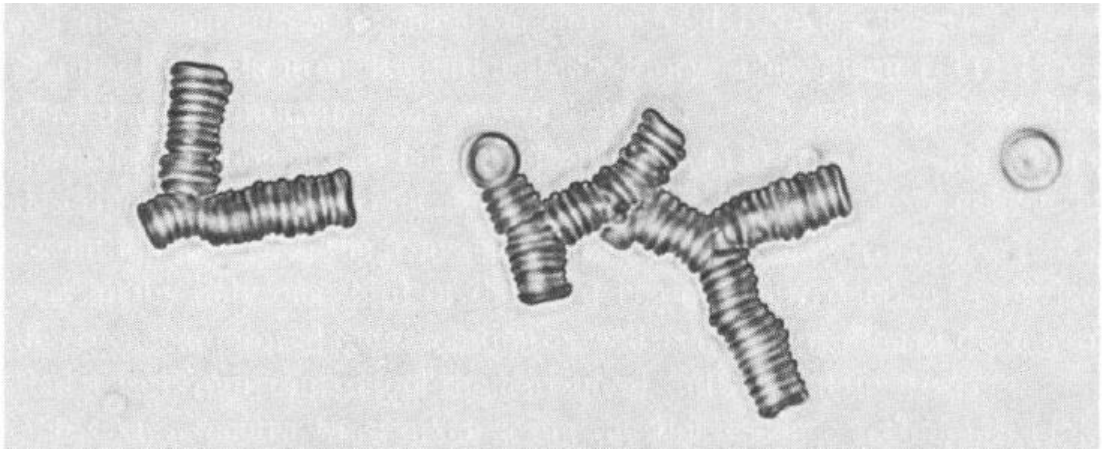

Figure 122: Picture of rouleaux formation found experimentally. Image from [49].

It has been found that the likelihood of contacting cells forming rouleaux is dependent on, among other things, time of contact [5]. Therefore, implementation of the rouleaux formation can be simply enacted when the appropriate *K_s* values are determined. With physically accurate *K_s* values, approximate time of contact is easy to determine, so fixing points to simulate sticking cells can be implemented in a relatively straightforward, though challenging, manner.

As of the writing of this thesis, these special cases for sedimentation and resting contact are not implemented into the contact model because it falls outside of the scope of that which is attempted to be accomplished in this project.

Section 5.4: Applicability and Sources of Error

The results presented in this section present data that is valid for a wide range of experiments that should be conducted to give experimental data to aid in validation of the model. Further development of this code should be conducted to handle the fluid-cell interaction that would be experienced during physical experiments. Clearly, the fluid interaction would cause a significant effect on the overall dynamics of the system. Although the code architecture is written in such a way that it may handle fluid nodes, the fluid interaction with other fluid and blood cells needs to be implemented in order to function properly. The implementation of the fluid node interaction falls outside of the scope of this project, as the author sought a method of appropriately handling cell-cell interaction.

The model is well-suited to deal with direct collisions because of the stability seen in the simulations. The images of the simulation from both the sedimentation and direct collision simulations qualitatively appear to accurately recreate the expected physics of two bodies colliding. These simulations are presented in a way that would make them easily validated with a simple experiment that visualizes red blood cells colliding in two dimensions. This visualization could be easily accomplished by manipulating one cell with attached glass microbeads as in [42]. The sedimentation simulations could also be simply validated by visualizing a cell sedimentation experiment as in [48] but with a high frequency camera. The rotated cell simulations are not immediately clear as to how applicable they can be for validation. First, the

amount of rotation experienced by these cells is heavily influenced by the location at which the cells collide. The vertical distance from the center of mass to the collision of the vertical cell is roughly equivalent to the distance of a moment equation M=F*d where F is the collision force. Changing this distance may have a significant effect on the rotation experienced by the red blood cell.

More importantly, the simulations highlight a necessity for improvement of the membrane damping to allow for rotations when there is no force being applied. Once the colliding cells lost contact, it was clear that the membrane damping removed the velocity of each node that accounted for rotation, quickly leading to the complete cancellation of rotation. This issue is not serious when the cells remain in contact as would be expected in a simple sedimentation simulation, so this inability to sustain rotation is not expected to cause inaccuracies for simulating large-scale sedimentation. However, in cases of single collisions as would be seen in flow through an artery or vein, the damping model must be modified to deal with the rotational damping problem. Until this can be accomplished, it will be considered a source of error in the rotational simulations.

Chapter 6: Conclusions

In this thesis, a new method of dealing with collisions based on the discrete element method has been proposed which attempts to accurately model the collisions of simulated red blood cells. The method was developed as part of a larger project which seeks to simulate red blood cells as they flow through an artery downstream of an artificial heart valve. This interpretation of the discrete element method sought to accurately recreate the dynamics of a colliding red blood cell as researchers have found that certain collisions can lead to spontaneous hemolysis [7, 8] and that artificial heart valves can cause an entrainment effect which may lead to a greater frequency of collisions.

The discrete element method was chosen for its ability to deal with collisions in time without needing to alter the time step as is the case in explicit or event-driven molecular dynamics. This ability became important in incorporating the collision model with the various time step integration techniques that are being used in the red blood cell models proposed by the groups that are collaborating on this project. The time step integrations that are being used often result in any particular node's position and velocity being calculated then later corrected based on the forces that the node experiences. For this reason, an event-driven model would significantly hinder the ability of the time step integration techniques to recreate accurate dynamics. Thus, a method that applies forces based on the collision geometry is desired to allow for a more gradual evolution of nodal positions for smoother and more physically correct results.

To be able to implement this method, the momentum of the nodes involved in the collision needed to be conserved when forces were applied. A direction of the force was proposed based on the location of each of the two red blood cell's center of mass such that the direction of the force is equal and opposite. The total force was then equally distributed between the two red blood cells, and then further divided equally between all nodes involved in the respective collision. This guarantees that the force applied for the collision is equal in magnitude and opposite direction for the whole system. By examining only the forces due to the collision, it is clear that this satisfies the conservation of momentum which is of utmost importance in discrete element method simulations. The model does not include a frictional force as of the time of writing of this thesis due to the non-trivial nature of determining not only the magnitude of the friction force, but also the direction in which this force should be applied.

In determining the magnitude of this collision force to be applied to the nodes, a measure of the collision geometry was necessary. This method was derived from the conclusion drawn from [27] that the area of overlap in 2-dimensional simulations is directly proportional to the force that should be applied to deal with said collision. In three dimensions, it was reasoned that the volume of overlap is a measure of deformation and, therefore, should be directly proportional to the collision force. Due to the unique nature of these colliding bodies, a new method for calculating volume of arbitrary discretized surfaces was needed. Several methods were proposed to test

which was most appropriate for this given application. The Principal Component Analysis Dot Product (PCADP) method was determined to be the most accurate that also met the efficiency requirements placed on the code. Once the volume of the collision was approximated, forces could be applied directly to intersecting nodes.

Simulations were carried out that attempted to qualitatively validate the model due a lack of experimental data. A range of spring constants, *K_s*, were used to be able to create a range of values that could be used for experimental validation. The *K_s* value is the coefficient used to determine the contact force based on the volume. The value *K_s*=75 was focused on as a value that provided what appeared to be reasonable accurate dynamics along with reasonable potential energy terms. For each simulation, the results were analyzed qualitatively to create a preliminary range of possible *K_s* values. Afterwards, the potential energy terms of each simulation were checked for continuity and smoothness to ensure that the simulations are physically realistic and stable. A direct collision simulation was conducted as the simplest type of collision that could be expected in vivo. In order to further test the method, an angled collision simulation was performed and in addition to the potential energy terms, the amount of rotation of the red blood cell was analyzed which highlighted both the need for incorporating a frictional force as well as for the revision of the damping model to allow for continued rotation even after contact between two cells is lost. Finally, a sedimentation collision simulation was performed in an attempt to display the effectiveness of this method in a worst-case scenario of an initial large jump in collision volume. The method again appears to produce realistic dynamics

between the two red blood cells. The idea of the effective coefficient of restitution (ECOR) was introduced (Equation 5.1) as the ratio of the cell velocities before and after collision. Energies and effective coefficients of restitution were also analyzed and it was found that the ECOR values were different with the same *K_s* values which gave rise to a hypothesis that the dynamics of a red blood cell are highly dependent on the orientation of collision. Due to the unique structure of red blood cells, experiments should be performed to either confirm or disprove the presence of this effect.

One of the areas that requires further development is the implementation of a fluid-cell interaction method. The dynamics of red blood cells are heavily dependent on the fluid flow of a vein, but development of this interaction falls outside of the scope of this thesis. Most important for the development of the code presented herein is the necessity for the improvement of the current damping model and its inability to handle cell rotation appropriately.

In conclusion, this interpretation of the discrete element method appears to be very effective in producing realistic dynamics while being efficient for use in a large-scale, highly parallelized code. This will allow for the expansion of the current code to be able to handle many blood cells and their interactions. This collision model is an important step in expanding the code to reach this end.

Chapter 7: Outlook

To continue the development of this method, several areas could be addressed. The first, and most pressing, is a necessary modification of the damping model used for red blood cell simulations. The collision model proposed in this thesis applies concentrated forces to specific nodes. These forces cause velocities that then propagate through the red blood cell, which causes change in the body's total velocity. However, when the forces applied cause a moment on the body, there is a relative velocity between each node due to rotation that is then damped by the current damping model. A specialized damping model may need to be developed to allow for free rotation after contact is lost between two bodies.

Second, the implementation of a frictional force is believed to dramatically improve dynamics of the system, particularly in a case such as the one in Section 5.2, where it appears as though the red blood cell surfaces slip along one another. Incorporating a frictional force would change the rotational effects in such a simulation by implementing a coefficient of friction. While it is understood that the coefficient of friction would depend on the relative velocity of the red blood cells, work should be done to determine appropriate directionality and magnitude [23].

Further refinement could be added to this model by accounting for resting contact of two cells. Resting contact is a problem that has been widely studied in various discrete element method codes [26, 24, 28]. Because the method applies repulsive forces to prevent extensive penetration, bodies are not able to remain in direct contact

for extensive periods of time. However, in many cases, these discretized bodies do remain in contact so codes are written to specially handle the cases in which resting contact would occur. In red blood cell applications, there have been studies that link contact time between cells to the ability to form rouleaux [49, 5], so the code should be specialized to allow red blood cells to bond together if a certain set of criteria are met.

A total body force, one in which a force smaller than the collision force is applied to the entire body, may be included to reduce collision time to allow for larger values of K_s while also giving more explicit control of the effective coefficient of restitution. This total body force could be proportional to the collision volume which would serve to reduce contact time, or depend on the relative velocities of the blood cells, which would act as a damping force similar to that defined in [23].

Chapter 8: Contributions

This thesis offers contributions that are useful in the implementation of the discrete element method. The most important contributions are:

1) The creation and validation of a volume calculation method that can create an accurate approximation based only on the coordinates that define the discretized surface
2) Development of a three-dimensional discrete element collision method that deals with highly deformable bodies that is able to be integrated with any time step integration technique
3) Qualitative visualizations of red blood cell collisions at various angles to define an effective range of appropriate *K_s* values to be used in future simulations
4) Presentation of a range of effective coefficient of restitution values under various situations to allow for simple comparison for validation by physical experiments

The development of this discrete element method code represents a significant step in the progress of the intention to simulate millions of red blood cells flowing freely through an artificial heart valve. Once validated, this method will allow for the use of any time step integration method because, instead of modifying the coordinates of a node directly, it applies forces that can then be handled by the integration technique. The code makes use of a volume approximation that may prove useful for simulations

in which a discretized surface's nodes are defined but its connectivity is not. Examples of such simulations are powder compaction simulations as seen in pharmaceutical and powder metallurgy simulations [36, 37]. The volume method allowed for the use of a general discrete element method application which can make use of any time step integration to allow for more realistic dynamics.

Additionally, this method was applied to red blood cell collisions and the results were visualized. These qualitative observations allow for the assumption of an appropriate spring coefficient value, *K_s*, until experiments can be conducted to validate the values herein. Quantitatively, a range of effective coefficients of restitution were presented as a value that is easily measurable experimentally. These simulations also presented the phenomenon that the effective coefficient of restitution changes based on the orientation of the collision. Experiments should be conducted to determine if this is an artifact of the method proposed or if this is a result of the irregular geometry and makeup of the red blood cell. In finding these values, an inherent weakness of the current damping model to handle rotational velocity of the red blood cell was highlighted as an area to be addressed to further the realism of the simulation results.

Appendix 1: Principal Component Analysis

Principal component analysis (PCA) is a statistical analysis method that is useful in determining vectors that correlate to the predominant directions of the points being examined [50]. The method, as was described in this work, will deal with N 3-dimensional points, where N is greater than or equal to 3. The results of the method can give up to N 3-dimensional vectors in the principal directions.

Because the method as it is used here is being used to determine the approximate shape of an ellipsoid, we will only consider the first three vectors given from PCA. To carry out the principal component analysis, simple matrix algebra can be conducted that is available in several major software packages. First, the data for which the principal components are being found needs to be normalized for the set's center of mass. In other words, the nodal coordinates being passed should have a center of mass located at the origin of the system so that we can give clear physical meaning to the resulting principal components. Starting with the matrix, Pos, which contains the nodal coordinates:

$$Pos_{ij} = Pos_{ij} - \langle Pos_j \rangle, \qquad i = 1, \dots N, \qquad j = 1,2,3 \tag{A.1}$$

The subscript *i* is the number of nodes contained within *Pos* while *j* is the dimensionality of the system. After normalizing the *Pos* matrix, the covariance matrix should be created to develop a 3x3 matrix which allows for the principal component analysis to output 3 principal directions as required for the ellipsoidal approximation.

$$Cov = \frac{1}{N-1} Pos^T Pos \tag{A.2}$$

The eigenvectors of the covariance matrix can then be found which correspond to the principal directions of the original *Pos* matrix. To find the eigenvectors of the covariance matrix, DGEEVX was used from the open source library LAPACK. These eigenvectors are then used to approximate volume as described in Section 4.3.

Appendix 2: FLASH Code Information

Overview

As has been mentioned earlier, the intention of the algorithm is to be able to apply it to a simulation in which millions of blood cells are allowed to flow freely through an open heart valve. To achieve this, the code needs to be written in such a way that each body stores its own data independently such that the code does not need to store global data which significantly slows down parallelized code. Instead, independent data storage requires less intercommunication between computer processors, leading to a large improvement in efficiency. The following sections will discuss how the code was organized and the order in which the subroutines are run and how the calls pass data between the main contact program and its subroutines.

Architecture

Each blood cell has a data structure associated with the bodies and nodes it is in contact with. Within this structure, there are four substructures used to store this data. First, an integer array `intersect_count` is created that has the dimension (numBodies). This array contains the number of nodes from the body that have intersected other bodies and saves this number in the dimension corresponding to the intersecting body. Another array, `intersectedNodes` with dimensions (numBodies, 300) stores the index of each of the intersecting nodes saved in the row corresponding to the intersecting body. The 300 is a dummy value that is large enough to ensure that there are never more nodes than spaces available in `intersectedNodes`. The final

two substructures store data related to collision volume and resultant force. The `intersectedVolume` is a 1-dimensional array of length (numBodies) where each entry is the volume between the master body and the body corresponding to the index of each row in the array. Similarly, the `intersectedForce` is a 1-dimensional array of length (numBodies) where each entry is the force due to the collision volume between the master body and the body corresponding to the index of each row in the array.

For example, in a 2-body problem, if nodes 145, 146, and 147 from body 1 are intersecting body 2, the vectors and matrices corresponding to body 1 would be:

$$intersectCount = \begin{Bmatrix} 0 \\ 3 \end{Bmatrix}$$

$$intersectedNodes = \begin{Bmatrix} 0, \dots 0 \\ 145, 146, 147, 0, \dots 0 \end{Bmatrix}$$

$$intersectVolume = \begin{Bmatrix} 0 \\ Vol\ between\ bodies\ 1\&2 \end{Bmatrix}$$

$$intersectForce = \begin{Bmatrix} 0 \\ Force\ between\ bodies\ 1\&2 \end{Bmatrix}$$

The contact model described above is implemented in the subroutine that deals with the integration of the advancement of the positions of the nodes on the surface of the blood cell. This was chosen to allow for easier retrieval of the velocities used in the integration that are essential in collision detection. Following this call, some initial checks are run to ensure that the body is only being checked by its master processor and fits the criteria for local point selection from 4.2.2. If these criteria are met, the bodies may be checked for intersection and marked accordingly if there is an intersection. From this point, the codes in both Fortran and Flash are nearly identical.

For brevity, further discussion on these methods is omitted. Refer to sections 4.2-4.4 for this discussion.

Implementation

Almost immediately upon attempting to develop with this contact model, we are faced with the problem of how to deal with blood cells that are close enough for contact to happen, yet belong to two separate processors. This problem arises from the desire to reduce inter-processor communication to maximize parallelization. Also, the need to send data reduces the available memory to a given processor which further reduces efficiency. As of the writing of this thesis, this problem is handled by ghost particles. These are particles that exist close to other bounding boxes and only the coordinates, velocities, and element data are passed to these adjacent bounding boxes to allow for collision detection. With this, collisions can be checked on a processor-by-processor basis. This also allows us to deal with particles outside of a processor's defined bounding box without needing to store all of the data belonging to the particular blood cell. The ghost particles are handled in exactly the same way as the particles actually belonging to the processor without needing to calculate the internodal force routines and time step integration. Collision detection, volume calculation, and collision force calculations do not rely on anything but the data contained in the ghost particles, and the resulting data from the contact model is passed back to the ghost particle's original processor to handle contact force integration in a parallel manner. Note that the collision detection only needs to happen between two cells just once. Therefore, some cells' collision detection will not be handled by its respective master processor.

Bibliography

[1] M. Heron, "Deaths: Leading Causes for 2010," *National Vital Statistics Reports,* vol. 62, no. 6, pp. 1-97, 2013.

[2] T. Yagi, D. Ishikawa, H. Sudo, T. Akutsu, W. Yang, K. Iwasaki and M. Umezu, "Real-Time Planar Spectral Analysis of Instantaneous High-Frequency Stress on Blood Cell Downstream of an Artificial Heart Valve," *12th International Symposium on Flow Simulation,* 2006.

[3] T. Yagi, W. Yang and M. Umezu, "Effect of Bileaflet Valve Orientation on the 3D Flow Dynamics in the Sinus of Valsalva," *Journal of Biomechanical Science and Engineering,* vol. 6, no. 2, pp. 64-78, 2011.

[4] E. Trowbridge, "The physics of arteriole blood flow. I. General theory," *Clinical Physics and Physiological Measurement,* vol. 4, no. 2, pp. 151-175, 1983.

[5] S. Kim, J. Zhen, A. Popel, M. Intaglietta and P. Johnson, "Contributions of collision rate and collision efficiency to erythrocyte aggregation in postcapillary venules at low flow rates," *American Journal of Physiology - Heart and Circulatory Physiology,* vol. 293, pp. 1947-1954, 2007.

[6] H. Ezzeldin, "Multi-scale Modeling of Soft Matter: Gas Vesicles and Red Blood Cells," University of Maryland, College Park, 2012.

[7] T. Yagi, S. Wakasa, N. Tokunaga, Y. Akimoto and M. Umezu, "Single-cell real-time imaging of flow-induced hemolysis using high-speed microfluidic technology," *IFMBE Proceedings,* vol. 25, pp. 2337-2340, 2009.

[8] T. Yagi, S. Wakasa, N. Tokunaga, Y. Akimoto and M. Umezu, "Collision Dynamics of Red Blood Cells Using High-Speed Impinging Microjets," in *International Conference on Fluid Control, Measurements, and Visualization*, Moscow, 2009.

[9] I. V. Pivkin and G. E. Karniadakis, "Accurate Coarse-Grained Modeling of Red Blood Cells," *Physical Review Letters*, pp. 1181051-1 - 118105-4, 2008.

[10] D. Fedosov, B. Caswell and G. Karniadakis, "Systematic coarse-graining of spectrin-level red blood cell models," *Computer Methods in Applied Mechanics and Engineering,* vol. 199, pp. 1937-1948, 2010.

[11] D. Fedosov, B. Caswell and G. Karniadakis, "A Multiscale Red Blood Cell Model with Accurate Mechanics, Rheology, and Dynamics," *Biophysical Journal,* vol. 98, pp. 2215-2225, 2010.

[12] D. Discher, N. Mohandas and E. Evans, "Molecular Maps of Red Cell Deformation: Hidden Elasticity and in Situ Connectivity," *Science,* vol. 266, pp. 1032-1035, 1994.

[13] D. Fedosov, *Thesis: Multiscale Modeling of Blood Flow and Soft Matter,* Brown University, 2007.

[14] P. Espanol, "A Fluid Particle Model," 1997.

[15] M. Giersiepen, L. Wurzinger, R. Opitz and H. Reul, "Estimation of shear stress-related blood damage in heart valve prostheses in-vitro comparison of 25 aortic valves," *International Journal of Artificial Organs,* vol. 13, no. 5, pp. 300-306,

1990.

[16] M. Grigioni, U. Morbiducci, G. D'Avenio, G. Benedetto and C. Del Gaudio, "A novel formulation for blood trauma prediction by a modified power-law mathematical model," *Biomechanics and Modeling in Mechanobiology,* vol. 4, no. 4, pp. 249-260, 2005.

[17] D. Arora, M. Behr and M. Pasquali, "A Tensor-Based Measure for Estimating Blood Damage," *Artificial Organs,* vol. 28, pp. 1002-1015, 2004.

[18] E. Evans, V. Heinrich, F. Ludwig and W. Rawicz, "Dynamic tension spectroscopy and strength of biomembranes," *Biophysical Journal,* vol. 85, no. 4, pp. 2342-2350, 2003.

[19] D. Frenkel and B. Smit, Understanding Molecular Simulation: from Algorithms to Applications, San Diego: Academic Press, 2002.

[20] E. Fermi, J. Pasta and S. Ulam, "Studies of non-linear problems," *LASL Report,* 1955.

[21] J. Gibson, A. Goland, M. Milgram and G. Vineyard, "Dynamics of radiation damage," *Physical Review,* vol. 120, pp. 1229-1253, 1960.

[22] A. Rahman, "Correlations in the motion of atoms in liquid argon," *Physical Review,* vol. 136, pp. A405-A411, 1964.

[23] C. Rapaport, The art of Molecular Dynamics Simulation, Cambridge: Cambridge University Press, 2004.

[24] P. Wriggers, Computational Contact Mechanics, Berlin: Springer, 2002.

[25] T. Kruppa, T. Neuhaus, R. Messina and H. Lowen, "Soft repulsive mixtures under gravity: Brazil-nut effect, depletion bubbles, boundary layering, nonequilibrium shaking," *The Journal of Chemical Physics,* vol. 136, 2012.

[26] T. Poschel, "Molecular dynamics of arbitrarily shaped granular particles," *Journal of Physiology,* vol. 5, pp. 1431-1455, 1995.

[27] H. Matuttis, S. Luding and H. Herrmann, "Discrete Element Simulations of Dense Packings and Heaps made of Spherical and Non-Spherical Particles," Institute for Computer Applications, Stuttgart, 1999.

[28] P. Wriggers, T. Vu Van and E. Stein, "Finite Element Formulation of Large Deformation Impact-Contact Problems with Friction," *Computers and Structures,* vol. 37, pp. 319-331, 1990.

[29] C. Boon, G. Houlsby and S. Utili, "A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method," *Computers and Geotechnics,* vol. 44, pp. 73-82, 2012.

[30] T. Poschel and T. Schwager, Computational Granular Dynamics, Berlin: Springer, 2005.

[31] A. Munjiza, The Combined Finite-Discrete Element Method, Chichester: John Wiley & Sons Ltd, 2004.

[32] K.-i. Tsubota, S. Wada, H. Kamada, Y. Kitagawa, R. Lima and T. Yamaguchi, "A Particle Method for Blood Flow Simulation, - Application to Flowing Red Blood Cells and Platelets," *Journal of the Earth Simulator,* vol. 5, pp. 2-7, 2006.

[33] J. Anderson, "HOOMD blue," University of Michigan, 2011. [Online]. Available: http://codeblue.umich.edu/hoomd-blue/. [Accessed 2013].

[34] M. Kleppmann, "Simulation of colliding constrained rigid bodies," University of Cambridge, Cambridge, 2007.

[35] J. Hallquist, G. Goudreau and D. Benson, "Sliding Interfaces with Contact-Impact in Large-Scale Lagrangian Computations," *Computer Methods in Applied Mechanics and Engineering,* vol. 51, pp. 107-137, 1985.

[36] G. Frenning, "An efficient finite/discrete element procedure for simulating compression of 3D particle assemblies," *Computer Methods in Applied Mechanics and Engineering,* vol. 197, no. 49-50, pp. 4266-4272, 2008.

[37] G. Frenning, "Compression mechanics of granule beds: A combined finite/discrete element study," *Chemical Engineering Science,* vol. 65, pp. 2464-2471, 2010.

[38] M. Griebel, S. Knapek and G. Zumbusch, Numerical Simulations in Molecular Dynamics, Berlin: Springer, 2007.

[39] T. Zohdi, An Introduction to Modeling and Simulation of Particulate Flows, Philadelphia: SIAM, 2007.

[40] A. Whitkin and D. Baraff, *Physically Based Modeling: Principles and Practice,* Carnegie Mellon University, 1997.

[41] F. Preparata and M. Shamos, Computational geometry: an introduction, New York: Springer-Verlag, 1985.

[42] M. Dao, C. Lim and S. Suresh, "Mechanics of the human red blood cell deformed by optical tweezers," *Journal of the Mechanics and Physics of Solids,* vol. 51, pp. 2259-2280, 2003.

[43] M. Dao, J. Li and S. Suresh, "Molecularly based analysis of deformation of spectrin network and human erythrocyte," *Materials Science and Engineering,* vol. 26, pp. 1232-1244, 2006.

[44] D. Discher, D. Boal and S. Boey, "Simulations of the Erythrocyte Cytoskeleton at Large Deformation. II. Micropipette Aspiration," *Biophysical Journal,* vol. 75, pp. 1584-1597, 1998.

[45] P.-O. Persson, "DistMesh - A Simple Mesh Generator in MATLAB," UC Berkeley, 2012. [Online]. Available: http://persson.berkeley.edu/distmesh/.

[46] E. W. Weisstein, "Spherical Cap," MathWorld - A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/SphericalCap.html.

[47] A. Pribush, D. Meyerstein and N. Meyerstein, "The mechanism of erythrocyte sedimentation. Part 1: Channeling in sedimenting blood," *Colloids and Surfaces B: Biointerfaces,* vol. 75, pp. 214-223, 2010.

[48] R. Smallwood, W. Tindale and E. Trowbridge, "The physics of red cell sedimentation," *Physics in Medicine and Biology,* vol. 30, no. 2, pp. 125-137, 1985.

[49] R. Samsel and A. Perelson, "Kinetics of Rouleau Formation," *Biophysical Journal,* vol. 37, pp. 493-514, 1982.

[50] I. Joliffe, Principal Component Analysis, Second Edition, Aberdeen: Springer,

2002.

[51] E. Wetzel and C. Tucker III, "Area Tensors for Modeling Microstructure During Laminar Liquid-Liquid Mixing," *International Journal of Multiphase Flow*, 1998.

[52] F.-J. Wang, L.-P. Wang, J.-G. Cheng and Z.-H. Yao, "Contact force algorithm in explicit transient analysis using finite-element method," *Finite Elements in Analysis and Design*, vol. 43, pp. 580-587, 2007.

[53] M. Heinstein, F. Mello, S. Attaway and T. Laursen, "Contact-impact modeling in explicit transient dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 187, pp. 621-640, 2000.

[54] C. Li, Y.-P. Liu, K.-K. Liu and A. Lai, "Correlations Between the Experimental and Numerical Investigations on the Mechanical Properties of Erythrocyte by Laser Stretching," *IEEE Transactions on Nanobioscience*, vol. 7, no. 1, pp. 80-90, 2008.

[55] R. Skalak and P.-I. Branemark, "Deformation of Red Blood Cells in Capillaries," *Science*, vol. 164, pp. 717-719, 1969.

[56] N. Bicanic, "Discrete Element Methods," in *Encyclopedia of Computational Mechanics*, Glasgow, John Wiley & Songs, Ltd., 2007, pp. 1-32.

[57] K.-i. Tsubota and S. Wada, "Elastic force of red blood cell membrane during tank-treading motion: Consideration of the membrane's natural state," *International Journal of Mechanical Sciences*, vol. 52, pp. 356-364, 2010.

[58] P. Maffettone and M. Minale, "Equation of change for ellipsoidal drops in viscous flow," *Journal of Non-Newtonian Fluid Mechanics*, vol. 78, pp. 227-241, 1998.

[59] S. Wakasa, T. Yagi, Y. Akimoto, N. Tokunaga, K. Iwasaki and M. Umezu, "Microscale Visualization of Erythrocyte Deformation by Colliding with a Rigid Surface Using a High-Speed Impinging Jet," *IFMBE Proceedings*, vol. 23, no. 2, pp. 1422-1425, 2008.

[60] R. Zhao, J. Antaki, T. Naik, T. Bachman, M. Kameneva and Z. Wu, "Microscopic investigation of erythrocyte deformation dynamics," *Biorheology*, vol. 43, pp. 747-765, 2006.

[61] Y. Kikuchi and K. Yomiyasu, "Red blood cell deformability and protein adsorption on red blood cell surface," Hokkaido University, Sapporo, 1984.

[62] N. Brilliantov, F. Spahn, J.-M. Hertzsch and T. Poschel, "The collision of particles in granular systems," *Physica A*, vol. 231, pp. 417-424, 1996.

[63] H. Kruggel-Emden, E. Simsek, S. Rickelt, S. Wirtz and V. Scherer, "Review and extension of normal force methods for the Discrete Element Method," *Powder Technology*, vol. 171, pp. 157-173, 2007.

[64] F. Tokumasu, G. Ostera, C. Amaratunga and R. Fairhurst, "Modifications in erythrocyte membrane zeta potential by Plasmodium falciparum infection," *Experimental Parasitology*, vol. 131, pp. 245-251, 2012.