ADHESIVE DYNAMICS SIMULATIONS OF TRANSPORT AND ADHESION OF CIRCULATING TUMOR MICROEMBOLI

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Kevin Jamahl Anderson

January 2017

ADHESIVE DYNAMICS SIMULATIONS OF TRANSPORT AND ADHESION OF CIRCULATING TUMOR MICROEMBOLI

Kevin Jamahl Anderson, Ph. D.

Cornell University 2017

Cancer metastasis is a multistep process through which tumor cells detach from the primary tumor and invade distant tissue sites. Studies have shown that tumor cells can invade as multicellular aggregates or clusters known as circulating tumor microemboli (CTM). CTM are potentially advantageous to individual cells in the survival, proliferation, and establishment of micrometastatic lesions in distant organs, and the presence of CTM in blood is seen as a marker of highly metastatic potential. In this thesis, a novel adaptation of the Multiparticle Adhesive Dynamics (MAD) simulation was developed to analyze the behavior of model CTM interacting with activated endothelium. The model CTM, based on the morphology of Colo205 cell line samples, consisted of doublet, triplet, and 4-mer aggregates in simple geometrical conformations. These aggregates, coated with sialyl Lewis^x, were applied to a series of hydrodynamic and adhesive simulations in the presence of an E-selectin-coated plane wall. Simulations consisting of the hydrodynamic component of the MAD simulation analyzed the effects of model aggregate conformation and orientation on adhesive binding potential. Larger CTM conformations with intermediate nonsphericities had the highest adhesion potential. Adhesive interactions were also evaluated, and *in vitro*

rolling assays were used to establish the MAD simulation as a predictor of CTM behavior in shear flow. Model CTM exhibited rolling and transient adhesion interactions under physiological conditions. The distribution of adhesion interactions observed was dependent on the aggregate size. Aggregates consisting of 3 cells exhibited the most stable rolling, and rod-like 4-mer particles were unable to form substantial tethers with the surface. The bond lengths and lifespans were measured, and the distribution of lengths and lifespans correlated with the types of adhesion interactions observed. The results of these simulations established this adaptation of the MAD simulation as a method to evaluate metastatic efficiency.

BIOGRAPHICAL SKETCH

Kevin Jamahl Anderson was born to Tommy and Cheryl Anderson on March 24th, 1990, in Heidelberg, Germany. He attended the University of Alabama at Birmingham, where he majored in Biomedical Engineering and was a member of the Science and Technology Honors Program. While there he conducted research for Dr. Yue Song, developing molecular dynamics simulations of poly-L-lysine interacting with neutral and negatively charged lipid bilayers to study the membrane translocation of cell penetrating peptides. In May 2011, Kevin earned a Bachelor of Science Degree in Biomedical Engineering with honors in Science and Technology, and in the fall he began his graduate studies in the Department of Biomedical Engineering at Cornell University. Kevin worked in the laboratory of Dr. Michael R. King, where he developed an adhesive dynamics simulation for the study of the transport and adhesion of circulating tumor cell aggregates. During the course of his studies Kevin has received the Sloan Foundation Fellowship, the Sage Diversity Fellowship, and the National Science Foundation Graduate Research Fellowship. After leaving Cornell, Kevin looks forward to continuing his career in biotechnology.

To Mom, Dad, and Nana, who have supported me more times and in more ways than I could ever write out on paper

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor, Professor Michael R. King. I know for some people, it was a daunting task to try to figure out which lab to commit to for 5-6 years, but the choice could not have been easier for me. When I first met him, I was inspired by the obvious passion he had for his work, and I felt right at home with the people in the lab. Dr. King has continued to be a great source of support, offering guidance and reassurance during the more challenging periods of my studies. I'd also like to thank my thesis committee members, Dr. Jeffrey Varner and Dr. Tracy Stokol, for their guidance and support. I really appreciate the advice and additional perspective that they have given towards my work. I'd also like to thank the department staff, particularly Belinda Floyd, who always worked out any problems or handled any questions whenever I had to figure something out.

I'd also like to thank my mother, father, grandmother, and brother, who put up with me skipping out on a lot of holidays. Unfortunately I was not close enough for easy visits, but when we did get together it was always a fantastic reminder of the love and support that helped me get to where I am today. I could not have done this without you.

A special shoutout goes to Dr. Andrew Hughes, Dr. Yue Geng, and Adelaide de Guillebon, and Dr. Weiwei Wang for their collaboration and extra help, in addition to being great people to work with. And to the rest of the lab mates and managers over the years, Dr. Jocelyn Marshall, Dr. Mike Mitchell, Thong Cao, Dr. Anne Rocheleau, Jeffrey Mattison, Dr. Siddharth Chandasekaran, Korie Grayson, Nerymar Ortiz-Otero, and Zeinab Mohamed, thanks for keeping the lab so simultaneously entertaining and

intellectually stimulating. Outside of lab, I have met some fantastic people in Ithaca, picked up a lot of fun new hobbies, and played on some great sports teams. Between the random adventures out on the commons, and the nights just hanging out getting to know each other, it has been a blast. And I'm glad that I was able to meet so many people that I could really connect with.

And last, but definitely not least, I'd like to thank my girlfriend Shirley. She has easily been the best thing to ever happen to me at Cornell. She has been a massive support, particularly during my last two years, and I can't thank her enough for all of the love, laughs, and motivation she has given me.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AD	Adhesive dynamics
AFM	Atomic Force Microscopy
BIE	Boundary integral equation
BSA	Bovine Serum Albumin
CAM	Cell Adhesion Molecule
CDL-BIEM	Completed Double Layer-Boundary Integral Equation Method
CTC	Circulating tumor cell
CTM	Circulating tumor microemboli
DPBS	Dulbecco's phosphate-buffered saline
ECM	Extracellular matrix
EGF	Epidermal growth factor
EMT	Epithelial to mesenchymal transition
EpCAM	Epithelial cell adhesion molecule
ESL-1	E-selectin ligand-1
FBS	Fetal bovine serum
FITC	Fluorescein isothiocyanate
GPIBα	Glycoprotein Ib platelet alpha subunit
HD-CTC	High-definition circulating tumor cell
IL-1β	Interleukin-1β
JNK	c-Jun N-terminal kinase
LAMP-1	Lysosomal membrane glycoprotein-1
LAMP-2	Lysosomal membrane glycoprotein-2
mAb	Monoclonal antibodies
MAD	Multiparticle Adhesive Dynamics

MAPK	Mitogen-activated protein kinase
MESF	Molecules of equivalent soluble fluorochrome
Mgat5	β1,6-N-acetylglucosaminyltransferase V
MUC-1	Mucin-1
MUC-2	Mucin-2
NRMSD	Normalized root-mean-square deviation
PBS	Phosphate-buffered saline
PMN	Polymorphonuclear neutrophil
PSGL-1	P-selectin glycoprotein ligand-1
RMSD	Root-mean-square deviation
RPMI	Roswell Park Memorial Institute
SD	Standard deviation
SEM	Scanning electron microscope
sLe^a	Sialyl $Lewis^a$
sLe^x	Sialyl $Lewis^x$
TICA	Time integral of contact area
TNF-α	Tumor necrosis factor-α

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 – BACKGROUND & INTRODUCTION

1.1 THE METASTATIC CASCADE

Cancer is a leading cause of death worldwide. In 2012 it was responsible for 8.2 million deaths, and cancer deaths are expected to increase to 13 million annually over the next two decades[1]. The major factor contributing to the mortality of cancer is cancer metastasis. Cancer metastasis is a multistep process through which tumor cells detach from the primary tumor and travel through the circulation to invade distant tissue sites. Tumors are defined in part by unregulated cell growth, and as the tumor expands it eventually strips the surrounding tissue of nutrients and oxygen. This results in necrosis and the subsequent release of proinflammatory signals that stimulate angiogenesis and extracellular matrix degradation in addition to the recruitment of stromal cells. Immune cells, cancer-associated fibroblasts, stem and progenitor cells, and vasculature forming cells help cultivate a dynamic microenvironment that enables tumor progression and invasion[2, 3]. Genetic mutations and growth factor-stimulated signaling within the tumor microenvironment drives some tumor cells to undergo a phenotypic switch, known as the epithelial to mesenchymal transition (EMT). Tumor cells lose expression of proteins such as E-cadherin, switch to a more mesenchymal phenotype, detach from the primary tumor and intravasate into lymphatic and, primarily, blood vessels[4-6]. It has been estimated that 1 million tumor cells per gram of primary tumor are shed daily[7]. Cells that enter the bloodstream and are able to avoid anoikis, apoptosis due to loss of cell-cell and cell-matrix adhesion, can travel through the bloodstream to distant tissue sites. These circulating tumor cells (CTCs) travel through the bloodstream until they extravasate to

distant tissue sites to form secondary tumors[5].

Certain types of cancers, including breast, lung, liver, prostate, and colorectal, have a higher potential to form metastases than others, and these metastases can be specific to a particular set of organs[4]. Regardless of the destination of invasion, the overall mechanism is the same, and many studies on general metastatic behavior focus on analyzing one or a few cancer subtypes. In my studies, I use observations made from a colorectal cancer cell line as an example for mechanisms of metastasis. Colorectal cancer is the third most common cause of cancer-related deaths in men and women in the US. In 2013, approximately 1,177,556 people were living with the disease, and in 2014 it was the cause of an estimated 50,310 deaths, and an estimated 136,830 new cases were diagnosed[8, 9]. Prognosis for colorectal cancer patients is highly dependent on the staging of the tumor, and the improvement of prevention and early detection methods has resulted in strong progress in reducing incidence and death rates over the years[9]. Patients with stage I colorectal cancer, in which the tumor is confined to the submucosa of the gastrointestinal tract, have a 5-year survival rate of approximately 90%. However, patients with stage IV cancer, in which metastases have formed, have a median survival of only 6 months. Currently, surgery with adjuvant chemotherapy is the only treatment for patients with metastatic colorectal cancer. Unfortunately, for most patients this is insufficient to completely remove the disease, and recurrences are expected within 5 years post-surgery[10, 11].

1.2 SELECTIN-MEDIATED ADHESION TO THE ENDOTHELIUM

The process through which cancer cells extravasate to the endothelium is similar to that which allows leukocytes to be recruited to sites of inflammation and infection. This process is mediated by highly specific receptor-ligand interactions that allow flowing leukocytes to be brought into contact with and firmly adhere to activated endothelial surfaces before penetrating the subendothelial matrix[12].

The first step of the adhesion cascade involves the selectin family of cell adhesion molecules (CAMs). There are three types of selectins: E-, P-, and L-selectin, which are calcium-dependent transmembrane glycoproteins that all bind to carbohydrate ligands and share a similar structure. They have an N-terminal lectin domain, an epidermal growth factor (EGF)-like domain, a variable number of consensus repeat domains (2, 6, and 9 for L-, E-, and P-selectin, respectively), a transmembrane domain, and a cytoplasmic tail. P-selectin is stored in α-granules of platelets and in Weibel-Palade bodies of endothelial cells, and can be translocated to the cell surface within minutes after stimulation by inflammatory mediators such as thrombin or histamine[13-15]. E-selectin is also expressed on the surface of endothelial cells, upon stimulation by inflammatory cytokines such as tumor necrosis factor-α (TNF-α), interleukin-1β (IL-1β), or lipopolysaccharide. However, rather than being stored within the cytoplasm like P-selectin, E-selectin must be synthesized, and peak expression is not typically seen until 4-6 hours in static conditions. L-selectin is constitutively expressed on almost all types on leukocytes, particularly on the tips of microvilli [14, 16].

All selectins bind in a calcium-dependent manner to carbohydrates that are largely sialylated, fucosylated, and/or sulfated, as well as other molecules including heparin sulfate glycosaminoglycans and glycosphingolipids. A common structure recognized by all selectins is the tetrasaccharide sialyl Lewis^{x} (sLe^{x}) and its isomer sialyl Lewis^{a} (sLe^{a}). sLe^{x} is the terminal component of glycoproteins and glycolipids found on circulating leukocytes and tumor cells, as well as some endothelial cells. sLe^{a}, however, is primarily found on tumor cells, and its expression is associated with tumor progression[17, 18]. P-selectin glycoprotein ligand-1 (PSGL-1) is an extensively studied glycoprotein decorated with sLe^{x}, and it is currently the only known ligand for all three selectins. It has a particularly high affinity for P- and L-selectin. PSGL-1 is expressed on leukocytes, hematopoietic progenitor cells, and platelets[18, 19]. Leukocytes can also bind to E- and P-selectin on endothelial cells via L-selectin, CD44, and E-selectin ligand-1 (ESL-1)[20]. Tumor cells such as colorectal cancer cells have been found to bind to E-selectin via lysosomal membrane glycoprotein-1 (LAMP-1) and LAMP-2 and mucins such as MUC-1 and MUC-2[21]. Colorectal cancer cells can also bind to P-and L-selectin via mucin-type glycoproteins[22].

The selectin-mediated adhesion cascade is illustrated in Fig. 1.1. Blood flow brings circulating cells in close proximity to the vessel wall, where the cell ligands can bind to selectins on the endothelial cell. Because of the rapid velocity of the flowing cell, contact time is short, necessitating fast binding kinetics for selectin-ligand bonds. Once the cell forms initial tethers with selectins, the cell velocity slows, and more selectin bonds are formed. The dissociation of bonds formed at the trailing edge of the

cell is balanced by new bonds forming at the leading edge of the cell, resulting in the cell rolling on the blood vessel wall. As the cell rolls, adhesion and chemokine receptors on the cell transduce signals that lead to the activation of integrins, another class of cell adhesion molecules[23].

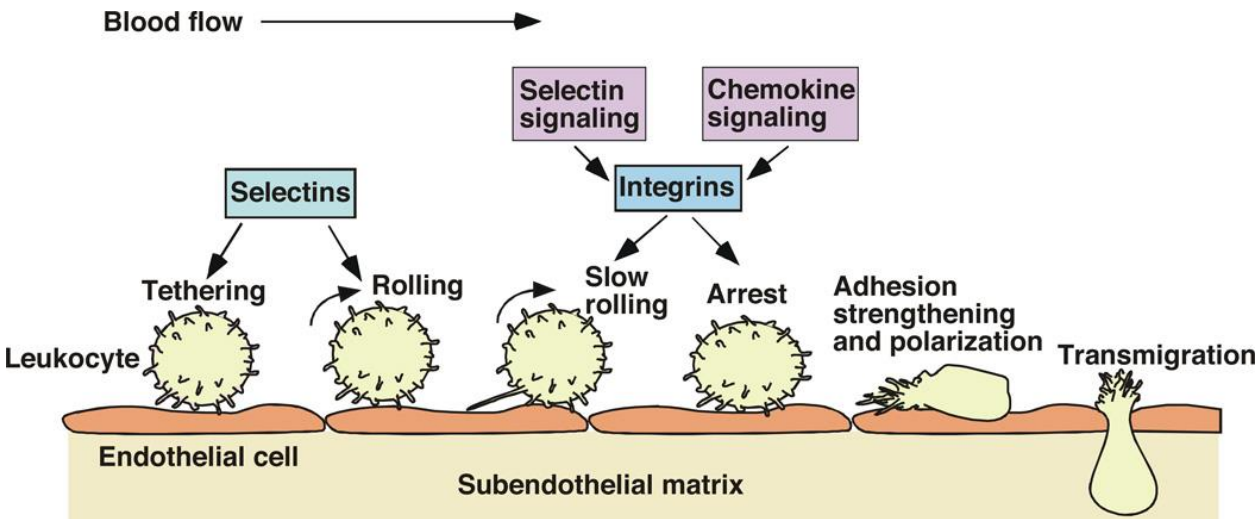

Figure 1.1: Selectin-mediated adhesion cascade. Selectins mediate tethering and rolling of leukocytes and tumor cells. Blood flow brings circulating cells in close proximity to the vessel wall, where receptors can bind to selectins on the endothelial cell. Once the cell forms initial tethers with selectins, the cell begins rolling on the endothelium. Rolling cells transduce signals that enable firm arrest and extravasation. Reproduced, with permission, from [23].

Integrins are large transmembrane glycoproteins consisting of an α-chain and β-chain subunit, and they are present on tumor cells, stromal cells, and components of the vasculature. Integrins can directly bind to components of the extracellular matrix (ECM), such as laminin, fibronectin, vitronectin, and collagen, facilitate cell motility and invasion. Integrins are linked to cytoskeletal structures, and when bound to ECM components can send signals bidirectionally to alter the conformation of the cell in response to the environment. Integrins also regulate angiogenesis, proliferation and survival[12, 24]. Integrins bind strongly to immunoglobin family transmembrane glycoproteins on endothelial cells, but have slower reaction kinetics, and as a result a study by Tandon and Diamond[25] found that integrins alone cannot facilitate cell adhesion in shear flow rates greater than 400 s^{-1}. Selectins are required to decelerate the rolling cell enough for integrins to activate and the cell to firmly arrest. In that same study, Tandon and Diamond found that aggregating neutrophils in shear flow required on average 52.5 ± 8.5 ms to form stable adhesions[25].

In addition, the selectins themselves can also modulate signal transduction upon binding. For example, E-selectin adhesive interactions result in the cytoplasmic domain linking with actin-associated proteins such as α-actinin, filamin, vinculin, FAK, and paxillin, leading to the activation of the mitogen-activated protein kinase (MAPK) pathway to support cell adhesion, migration, and the inflammatory response. The MAPK pathway mediates, among other processes, cell motility, shape change, and cell junction permeability, as well as the upregulation of genes responsible for the production of pro-inflammatory cytokines. Similarly, L-selectin binding induces the

activation of the MAPK and c-Jun N-terminal kinase (JNK) signaling pathways in leukocytes. With selectin-mediated cell rolling of circulating tumor cells, selectin and integrin binding between rolling cells and endothelial cells induces “outside-in” signaling within both cells, leading to activation of kinase cascades and cytoskeletal reorganization, allowing for arrested cells to migrate between or through endothelial cells into the underlying tissue to form a secondary tumor. [23, 26, 27].

1.3 METASTASIS OF CIRCULATING TUMOR MICROEMBOLI

In addition to the presence of CTCs in the bloodstream, circulating tumor microemboli (CTM) have been found in the bloodstream of colorectal, renal, prostate, lung, and breast cancer patients[5, 28]. These CTM can result from clusters of cells detaching from the primary tumor and migrating collectively into the bloodstream through leaky blood vessels characteristic of the tumor microenvironment[29]. CTM can also be formed intravascularly; tumor cells attached to the endothelium have been found to recruit other rolling and floating tumor cells and proliferate. Although extravasation of individual cells can occur, CTM can also outgrow the blood vessel they are adhered to and destroy it, resulting in metastasis without extravasation[30, 31].

Circulating tumor cell aggregation is mediated in part by galectin-3, a lectin that recognizes the β-galactoside carbohydrate group. The enzyme β1,6-N-acetylglucosaminyltransferase V (Mgat5) is overexpressed in cancer cells and catalyzes the synthesis of β-1,6-N-acetylglucosamine-branched glycoproteins and glycolipids, of which Galectin-3 has a high affinity for. This interaction mediates the

binding of galectin-3 with MUC-1, LAMP-1, and LAMP-2, glycoproteins overexpressed in cancer cells. β-1,6-N-acetylglucosamine can also be attached to cadherins, integrins, and other cytokine and growth factor receptors implicated in EMT.[32] One characteristic of EMT is the loss of E-cadherin expression accompanied by the upregulation of N-cadherin expression. Galectin-3 binding to N-cadherin, a mesenchymal cadherin, leads to the destabilization of cell-cell junctions, allowing for a more migratory cell phenotype. Additionally, galectin-3 promotes cancer cell aggregation through galectin-3/MUC-1 bonds between adjacent tumor cells. Galectin-3 also binds to desmoglein, a cadherin found on desmosomes, further promoting homotypic cell-cell adhesion[32, 33]. This cell-cell adhesion allows CTM to better retain their glycocalyx and other extracellular matrix components and avoid anoikis, a form of apoptosis that occurs due to the loss of cell-cell and cell-extracellular matrix adhesion[34].

Anoikis, apoptosis as a result of a lack of cell-ECM attachments, is thought to be a major factor in the inefficiency of CTCs to form secondary metastasis. Experimental studies have shown that only about 0.01-0.02% of cells injected into the bloodstream of mice were capable of forming metastases, and a rapid loss of viable CTCs is seen after first introduction into the bloodstream[35]. In a study testing the viability of metastatic rat embryonic cells in circulation, most of tumor cells underwent apoptosis within 24-48 hours[36].

In addition to resisting apoptosis, CTM are thought to have a survival advantage by the production of autocrine proinflammatory cytokines and matrix

proteases. The CTM structure may also serve as a protective barrier for the innermost cells from host immunological cells[29]. In addition to homotypic aggregation, circulating tumor cells have also been found to form aggregates with activated platelets, and this interaction is seen to increase evasion from natural killer cells as well[12]. Evidence of the survival and prometastatic advantage of CTM is found in early studies of CTCs and CTM; it has been shown that CTM form metastases more readily than the same amount of single tumor cells injected into the circulation[37].

1.4 MODELING CELL ADHESION

Advances in our understanding of the mechanisms underlying cancer growth and metastasis have led to promising developments in cancer detection and treatment methods in recent years. These developments have led to an increase in the 5-year relative survival rate for all diagnosed cancers from 49% during 1975-1977 to 69% during 2005-2011[38]. However, despite these advances cancer metastasis remains the cause of approximately 90% of all deaths in cancer patients[39]. Much attention has been given towards modeling the multiple steps of the metastatic cascade, as discoveries could potentially lead to the development of new anti-metastatic drugs. Inhibiting the adhesion and/or transendothelial migration of metastatic cells to prevent the formation of secondary metastases would greatly increase cancer patient outcomes[40].

Many studies of the metastatic cascade have particularly focused on the processes from intravasation leading up to extravasation, as extravasation is the crucial point that leads to the formation of secondary metastases. Conventional studies of metastasis

have been mostly limited to *in vivo* mouse models, which provide a platform to screen genes involved in the metastasis for specific organs, identify proteins that mediate tumor cell invasion, and determine the roles of the chemical factors and signaling mechanisms that mediate the various steps in the cascade[41-47]. However, with these *in vivo* models it is difficult to perform tightly regulated studies, and limited quantitative results can be obtained[48]. *In vitro* models such as microfluidic devices offer a popular alternative, as they allow for the establishment of regulated environments with finely tunable parameters. Microfluidic devices can be used to study specific adhesive events within *in vivo*-like, organ-specific microenvironments using human cell types. These devices can have embedded endothelial cells, or they can be coated with the components of endothelial cells that mediate adhesion events, like CAMs, in order to study isolated mechanisms that contribute to the metastatic process. Environmental conditions present *in vivo*, including biochemical gradients, flow profiles, and shear stresses, can be precisely controlled, and high quality, real-time images can be obtained. In addition, quantitative results can be readily collected. By compartmentalizing metastasis into its compositional steps and studying the effects of isolated interactions, we can develop a greater understanding of the process as a whole and identify correlations between environmental conditions and extravasation efficiency[40, 49].

Leukocyte extravasation has been more extensively studied than tumor cell extravasation, and due to their similarities, studies of leukocyte extravasation have often served as the basis for metastatic studies. Studies of E- and P-selectin-deficient

mice confirmed that the presence of endothelial selectins is necessary for leukocyte rolling, and therefore extravasation[50-53]. Several *in vitro* studies[54-60] expanded on the role of endothelial selectins in leukocyte rolling by perfusing dilute suspensions of leukocytes through microscale flow devices containing cultured endothelial cells or purified selectins. Other studies[61-63] used microbeads coated with selectin ligands as a replacement for leukocytes. In the studies with microbeads rolling behavior similar to that seen in neutrophils was observed, suggesting that the rolling behavior is driven by the biomechanical properties of the selectin-ligand interaction rather than cellular functions. However, leukocytes roll more steadily and with a slower velocity than microbeads at a given shear stress and substrate density. This was attributed to leukocyte microvilli, which extend during tether formation and increase bond lifetimes. In addition, the use of isolated molecules on microbeads enabled the determination of which surface proteins are sufficient for rolling, as well as the identification of critical components of said proteins. The *in vitro* studies of both leukocytes and microbeads quantified the proportion of samples that experienced adhesive events, and they measured instantaneous cell rolling velocity, tether length and tether duration in order to estimate kinetic parameters for selectin-ligand binding. These studies were able to observe the stochastic nature of leukocyte rolling. Rolling behavior is a function of a variety of factors, including time, fluid shear stress, and the density of selectins on the substrate. L-selectin-mediated rolling can only occur above a threshold shear level, and all selectins display what is known as catch-slip behavior. Initial calculations by Bell predicted that an increase in applied force would decrease

bond lifetimes, however at an intermediate range of applied force, increased force actually increases bond lifetimes. Bonds that display this behavior are known as catch bonds. At higher applied forces, catch bonds transition to slip bonds, where increased force decreases bond lifetimes[64-66]. For a given set of environmental conditions, observations of both leukocytes and microbeads have shown that selectin-mediated rolling is noisy; there are substantial fluctuations in instantaneous velocity and the frequency and duration of pauses in motion. While rolling on all selectins display qualitatively similar trends, mean rolling velocities and the shear stress ranges where catch and slip bonding occurs vary between selectins and for different selectin-ligand pairs[55, 58, 63, 67, 68]. *In vitro* studies of this nature were also performed for breast, colorectal, lung, and prostate cancer cell line suspensions perfused through flow devices coated with purified selectins or activated endothelial cells. In addition to observing the rolling behaviors seen with neutrophils, these experiments allowed for the identification of selectin ligands prominently expressed in cancer cells[69-75].

While *in vitro* flow assays are useful for quantitatively studying rolling behaviors, there are some limitations. The use of *in vitro* flow assays to study CTC rolling and adhesion enables the identification of the differences in the behavior of neutrophils and cancer cells, as well as the differences between different cancer cell types. Particularly, the use of these studies to calculate kinetic parameters such as the reaction rate of bond association or disassociation enables quantitative comparison between their selectin-ligand bonding behaviors. However, the accuracy of measurements for instantaneous and average rolling velocity, the most common

metrics analyzed, is limited by the resolution and frame rate of camera used. This resolution is often on the microscale, the same time and length scale of some tethering events. In addition, studies analyzing bond lifespans typically use artificially low substrate densities in order to assume that pauses in cell motion are mediated by single bonds. [55, 59, 76]. Computational models address these limitations, as direct observations can be made for physiologically relevant parameter ranges on time and length scales much smaller than possible with physical experiments. This could potentially lead to the discovery of tethering mechanisms or other subtle behaviors not found in *in vitro* studies. Computational models are also advantageous in that experimental variables can be more rigorously controlled, minimizing factors that would confound results[77-79]. The adhesive dynamics simulation developed by Hammer and Apte, for example, was used to systematically vary bond association rate, bond elasticity, and wall shear stress in order to develop state diagrams of leukocyte adhesive behaviors illustrating the effects of bond properties or shear flow on adhesive interactions for all three selectins[79].

Though the aforementioned studies have analyzed the mechanics underlying selectin-mediated rolling, and several other studies[80-83] have highlighted the utility in exploiting the selectin-mediated rolling process to develop diagnostic or treatment methods for cancer cells, these experiments largely focused on individual CTCs. Despite the relevance of CTM in metastasis formation and growth, research studying its behavior has been limited. This may have been due, in part, to an underestimation of the significance of CTM in the extravasation process. Traditionally, the main

mechanism for CTM formation of micrometastases was thought to be due to CTM lodging in capillaries due to their size; they were assumed to not intravasate. Recently, however, Au et al.[84] provided evidence that CTM can transit through small capillaries, suggesting that their importance may be greater than previously assumed. Efforts to study CTM have increased[85], but there are still few computational studies that have analyzed the behavior of CTM. Au et al. developed a numerical model of deformable CTCs compressing through capillaries, in tandem with their experimental data[84]. This model accurately described cell-cell interactions and predicted breast cancer and melanoma cell aggregation and disassociation during transit through small capillaries. Aggregates were found to natively adopt spherical conformations and then reorganize into single file chains when entering capillaries, returning to the spherical conformation upon leaving the capillary. The Newton lab has developed a numerical model of thrombin generation by CTCs to study coagulation as a result of CTC aggregation[86, 87]. Finite element models of lung cancer aggregates were placed at the openings of model vessel bifurcations in the presence of a thrombin concentration gradient, and cancer aggregation and vessel occlusion were observed. Larger aggregates were observed to more readily occlude blood vessels, and larger aggregates could be slowed by the bifurcation, resulting in slower velocity field and increased thrombin generation. Puliafito et al. also developed a numerical simulation that modeled aggregation of tumor cells[88]. Model pairs of prostate cancer cell line clusters were placed in proximity to each other in the presence a concentration gradient of a diffusible chemoattractant, and the speed of aggregation was measured. Clusters in

proximity were found to migrate towards each other and form larger aggregates. Aggregation rates were found to be independent of initial cluster size, and rates increased with increased chemoattractant concentration and decreased as cluster size increased. Rejniak adapted the Immersed Boundary Method developed by Peskin to model two dimensional deformable CTCs capable of homotypic adhesion and interactions with a model endothelium. Through variation of cell stiffness, he illustrated how changes in stiffness influenced the survival of cells subjected to blood flow and their efficiency in binding to and rolling on the endothelium. CTC clusters were able to survive a given fluid shear stress longer than individual CTCs, and their adhesion efficiency was enhanced[89].

These models successfully replicate the formation and maintenance of CTC aggregates in flow, but they are lacking in information on the interaction of aggregates with the endothelium. Aggregate rolling was observed in some studies, but aside from the Rejniak model, these models only draw distinction between rolling and flowing with no endothelial interaction. These models do not denote the different adhesion states that occur, such as tethering or firm adhesion. The Rejniak model did observe rolling and firm adhesion, but the model was only in two dimensions, and it and the other models only describe bonding behavior indirectly. The model developed by Au et al. represented the interactions between cells as short-ranged potentials, rather than any physical representation of bonds. The Newman lab model did not include any adhesive or interparticle interactions; cluster motion was driven solely by fluid flow. The model developed by Puliafito et al. also did not include any adhesive or

interparticle interactions. The model developed by Rejniak did include adhesive and interparticle interactions, but they were simplified interactions not based on any particular receptor-ligand kinetics[84, 86-89]. The most suitable model to study the interaction of CTM with the endothelium would ideally include mechanisms describing aggregate motion as well as specific analysis of the adhesion molecules mediating adhesive interactions.

1.5 THE MULTIPARTICLE ADHESIVE DYNAMICS SIMULATION

One computational model that can be adapted for the study of the interaction of CTM with the endothelium is the multiparticle adhesive dynamics (MAD) simulation, which is an improved version of the adhesive dynamics simulation. The adhesive dynamics (AD) simulation was developed by Hammer and Apte as a numerical simulation of a rigid sphere with randomly distributed surface ligands interacting with a receptor-coated plane surface under viscous shear flow. This model was novel in that it could describe the wide range of adhesive phenomena displayed by leukocytes in postcapillary venules[78]. The model originally described the interaction of L-selectin coated leukocytes binding to an E-selectin coated surface, but was adapted to describe leukocyte rolling via PSGL-1/P-selectin, sLe^x/E-selectin, and L-selectin/sLe^x interactions. The MAD simulation, created by King and Hammer, was an adaptation of the original AD model that included multiple particles to observe particle-particle interactions in addition to cell adhesion to surfaces. Aside from leukocyte behavior, the MAD simulation has also been utilized to model GPIbα/von Willebrand factor-

mediated platelet rolling[77, 90-94]. The MAD simulation consists of two interconnected components: a hydrodynamic calculation of the mobility of a rigid sphere in low Reynolds shear flow computed using the Completed Double Layer-Boundary Integral Equation Method (CDL-BIEM) and a Monte Carlo based adhesive dynamics simulation of receptor-ligand bond dissociation assuming Bell model reaction kinetics[91, 92]. The MAD simulation is uniquely suited for the study of rolling adhesion. This simulation is a multiscale model, capable of describing interactions on the molecular and cell level. Simulation time and length scales can be as low as $O(10^{-12})$, enabling accurate and comprehensive measurements of bond and cell motion and bond lifespans. Given an optimized parameter set, the simulation can predict the full range of tethering events observed *in vitro* and *in vivo*. In addition to the simulation results matching observations from physical experiments, the parameter values themselves have physiological relevance, as almost all were derived from measurements of the receptor-ligand pair being studied. A schematic of the MAD simulation is shown in Fig. 1.2.

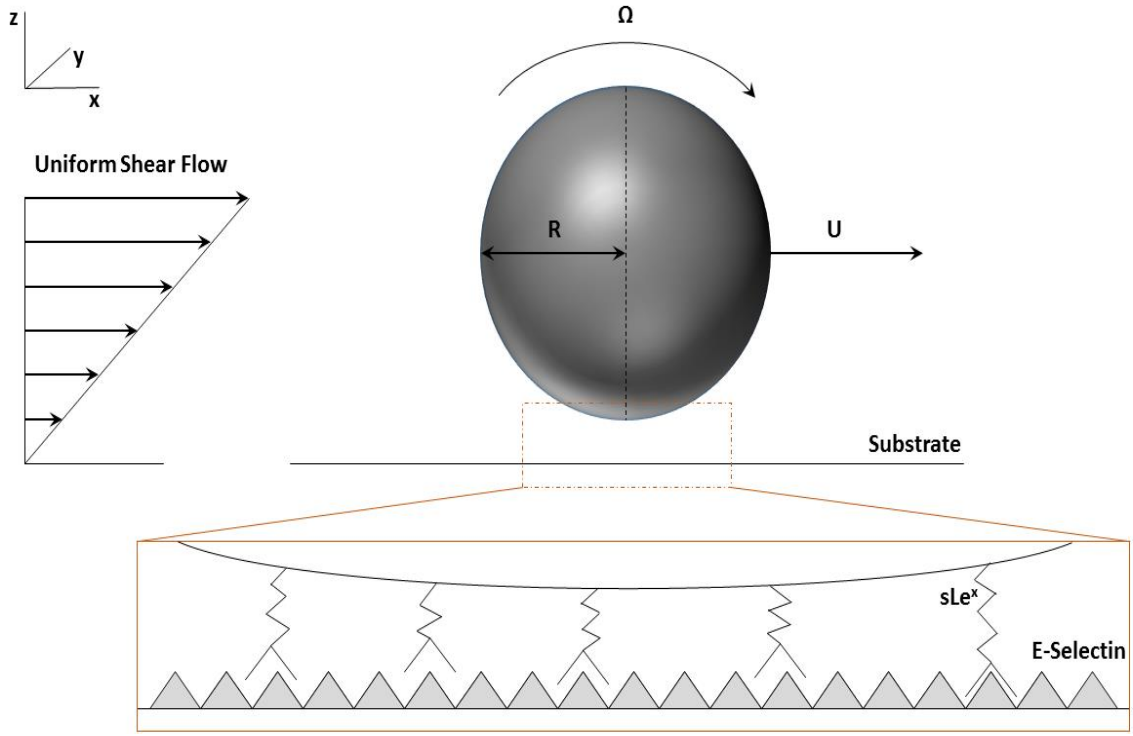

Figure 1.2: Schematic diagram of Multiparticle Adhesive Dynamics simulation. Linear shear flow brings the particle within reactive distance of E-selectin bound to the substrate. Once this reactive distance is reached, bond probabilities are calculated for bond association and dissociation using the Monte Carlo simulation. Bonds forming on the leading end of the cell are balanced by bonds breaking at the trailing edge of the cell, allowing it to roll. Instantaneous rolling velocity is governed by changes in the load balance between bonds on the leading and trailing cell edges.

Hydrodynamic calculation using the CDL-BIEM is described in detail by Kim and Karilla[95], so a brief overview will be given here. Physiological shear flow in postcapillary venules (0.8-4 dynes/cm^2 shear stress), is well within the Stokes flow regime[30, 78], so the Stokes approximation of the Navier-Stokes equations can be used to describe the motion of the fluid,

$$-\nabla p + \mu \nabla^2 \boldsymbol{u} = 0, \qquad \nabla \cdot \boldsymbol{u} = 0 \tag{1.1}$$

Where p is the pressure, μ is the fluid viscosity, and $\boldsymbol{u}$ is the velocity. Flow is assumed incompressible. The ambient flow is linear shear flow,

$$\boldsymbol{u}_\infty = Gz \tag{1.2}$$

where G is the shear rate and z is the distance from the wall. A no-slip condition is given at the surface of the wall, so the velocity is zero.

$$\boldsymbol{u}_0 = 0 \tag{1.3}$$

No-slip conditions are also given at the surface of each of the N particles,

$$\boldsymbol{u} = \boldsymbol{U}_\alpha + \boldsymbol{\omega}_\alpha \times (\boldsymbol{x} - \boldsymbol{x}_\alpha) \qquad \boldsymbol{x} \in S_\alpha \tag{1.4}$$

where $\boldsymbol{U}_\alpha$ and $\boldsymbol{\omega}_\alpha$ are the translational and rotational velocities of particle α, respectively, and $\boldsymbol{x}_\alpha$ and S_α are the center of mass and surface of particle α. The motion of an isolated cell is related to the forces acting on it by the 6 × 6 mobility matrix **M**:

$$\boldsymbol{u} = \boldsymbol{M}\boldsymbol{f} \tag{1.5}$$

where $\boldsymbol{u}$ is a six-element vector containing the particle's translational and rotational velocities, and $\boldsymbol{f}$ is a six-element vector containing the three components of net force and torque acting on the particle. The expanded mobility matrix relation is given in (1.6):

$$\begin{pmatrix} V_x \\ V_y \\ V_z \\ \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix} = \begin{pmatrix} \boldsymbol{M_{VF}} & \boldsymbol{M_{\Omega F}} \\ \boldsymbol{M_{VT}} & \boldsymbol{M_{\Omega T}} \end{pmatrix} \begin{pmatrix} F_x \\ F_x \\ F_x \\ T_x \\ T_y \\ T_z \end{pmatrix} \tag{1.6}$$

where V_x, V_y, and V_z are the Cartesian components of linear velocity, Ω_x, Ω_y, and Ω_z are the Cartesian components of angular velocity, F_x, F_y, and F_z are the Cartesian components of force, and T_x, T_y, and T_z are the Cartesian Components of torque. The mobility matrix relates the components of force and torque with the components of linear and rotational velocity, as shown by the 3×3 submatrices $\mathbf{M_{VF}}$, $\mathbf{M_{\Omega T}}$, $\mathbf{M_{VT}}$, and $\mathbf{M_{\Omega T}}$. For an isolated sphere near a plane wall in Stokes flow, all of the components of the mobility matrix are known[96-98], so if the net force and torque components acting on the particle are summed the instantaneous particle velocities can be determined from Equation 1.5. The integral representation of the Stokes equation is

$$u_j(\boldsymbol{X}) + \int_S H_{ij}(\boldsymbol{x},\boldsymbol{X}) u_i(\boldsymbol{x})\, dS(\boldsymbol{x}) = \int_S G_{ij}(\boldsymbol{x},\boldsymbol{X}) t_i(\boldsymbol{x})\, dS(\boldsymbol{x}) \tag{1.7}$$

where $H_{ij}(\boldsymbol{x},\boldsymbol{X})$ is the ith component of the stress tensor of G_{ij}, G_{ij} is the singularity solution resulting from a point force in the vicinity of a plane at $\boldsymbol{x}$ in the jth direction, $t_i(\boldsymbol{x})=\sigma_{ki} n_k$ is the stress tensor at $\boldsymbol{x}$, and $\boldsymbol{n}$ is the outwardly directed unit normal on S. The integral on the right side of the equation is called the single-layer potential, and the integral on the left hand side of the equation is called the double-layer potential. Equation 1.7 is a boundary integral equation (BIE) when $\boldsymbol{X}$ is evaluated at the particle surface ($\boldsymbol{X} \in S$). A BIE containing only the single layer potential is sufficient to represent the Stokes flow problem for rigid body motion of particles, but this equation

takes the form of a Fredholm integral equation of the first kind, with is generally ill-conditioned for a mobility problem. To circumvent this issue, the BIE containing the double layer potential is used. However, the double layer potential in itself is incapable of exerting any force or torque on the fluid, resulting in an indeterminate system containing null solutions. To complete the system, the forces and torques acting at the center of each particle are coupled with the null solutions on the particle surface to produce $6N$ additional linearly independent equations needed to make the double layer BIE fully determinate. The rigid body motions of each particle can then be extracted from the solution of the BIE and used to update the nodal points on each particle surface, centroid of each particle, and bond endpoints[91, 93].

Both the particle and the plane surface are coated with a steric layer to model the natural roughness of the cell glycocalyx layer. This surface roughness layer follows the assumption that both surfaces are covered with a bumpy region dense enough to cover the cell, but dilute enough for fluid to pass through. As a result this surface roughness layer is considered to have a negligible impact on the hydrodynamic calculations. The adhesive and repulsive interactions are exerted on the tips of these roughness elements. The surface roughness layers are included to compensate for lubrication forces preventing a mathematically smooth sphere from contacting a mathematically smooth plane[91, 93].

A very short-range, general repulsive force equation is included in the model to account for nonspecific interactions such as electrostatic repulsion:

$$F_{rep} = F_o \frac{\tau\, e^{-\tau\varepsilon}}{1-\tau\, e^{-\tau\varepsilon}} \tag{1.8}$$

where F_o is the magnitude of the repulsive force, $1/\tau$ is an interaction length scale on the order of angstroms, and ε is the separation distance between the two surface roughness layers. The repulsive force is directed normal to the plane wall. F_o and τ are experimentally determined to provide a force strong enough to prevent particle-wall overlap while allowing the surfaces to come within reactive distances of each other[91, 93]. An exponential relation of similar form was used by Bell, et al., and this general form is used in Stokesian dynamics simulations[99, 100].

The Monte Carlo simulation of receptor-ligand binding is described below. When the particle reaches reactive distance to the surface, a random distribution of ligands are generated on the surface of the particle in the contact area and tested for the probability of bond formation and breakage as described in Hammer and Apte[78]. The probability of bond formation P_f is tested using the relation

$$P_f = 1 - \exp\left(-k_f \Delta t\right) \tag{1.9}$$

where Δt is the timestep of the simulation and k_f is the forward rate of bond formation. Bonds formed are modeled as Hookean springs; the force applied to a bond is proportional to its deviation from equilibrium length. The equation for the forward rate of bond formation k_f is as follows:

$$k_f = k_f^o v_s \exp\left(\sigma|x_b - \lambda| \frac{\gamma - 0.5|x_b - \lambda|}{k_B T}\right) \tag{1.10}$$

where k_f^o is the intrinsic or unstressed on-rate, or the rate of association at zero applied force, v_s is the slip velocity between the surface of the particle and plane wall, σ is the Hookean spring constant, or bond stiffness, x_b is the bond length, λ is the equilibrium

bond length, k_B is the Boltzmann constant, and T is absolute temperature[94, 101]. An increase in slip velocity has been found to initially increase the forward rate constant, as the increase in relative velocity causes increased collisions between cell and surface ligands. Further increase in slip velocity results in a plateau of the forward rate constant, however, as the increased collisions are counterbalanced by the decreased contact time[102] The form of the k_f equation is derived from the Boltzmann distribution for binding affinity, and it is such that the rate of bond association is highest when the bond length is at equilibrium. To test an unbound molecule for bond formation, a P_f value is calculated, and then a random number is generated. If that random number is less than P_f, a bond is formed at the end of the time interval[91].

Any existing bonds are then tested for the probability of bond dissociation P_r, which has the same form as Equation 1.9:

$$P_r = 1 - \exp(-k_r \Delta t) \tag{1.11}$$

Where k_r is the reverse rate of bond dissociation. This version of the MAD simulation uses the widely accepted Bell model of bond dissociation kinetics, where an increased in applied force decreases bond lifespans. The equation for the reverse rate of bond dissociation k_r is as follows:

$$k_r = k_r^o \exp\left(\frac{\gamma \cdot \sigma |x_b - \lambda|}{k_B T}\right) \tag{1.12}$$

where k_r^o is the intrinsic or unstressed off-rate, or the rate of dissociation at zero applied force, γ is the reactive compliance, σ is the bond tensile strength, x_b is the bond length, λ is the equilibrium bond length, and k_bT is the product of the Boltzmann

constant and absolute temperature. The reactive compliance is a measure of the susceptibility for a bond to rupture under applied force; a smaller value means that the bond is less susceptible to rupture by force. Under the Bell model, as the applied force $F_b = \sigma|x_b\text{-}\lambda|$ increases on a bond, the bond lifetime decreases. This type of bond is referred to as a slip bond. For Colo205/E-selectin binding, the range of applied force where or slip bonding occurs was determined by *in vitro* flow assays to be wall shear stresses greater than 0.3 dyn/cm^2, or ≈150 pN of force[66]. To test a bound molecule for bond dissociation, a P_b value is calculated, and then a random number is generated. If that random number is less than P_b, a bond is broken during the time interval. Once all bonds have been tested, bond forces are calculated and the cell and bond positions are updated according to the kinematics of cell motion[65, 91, 92]. The solution algorithm for the MAD simulation is summarized in Fig. 1.3.

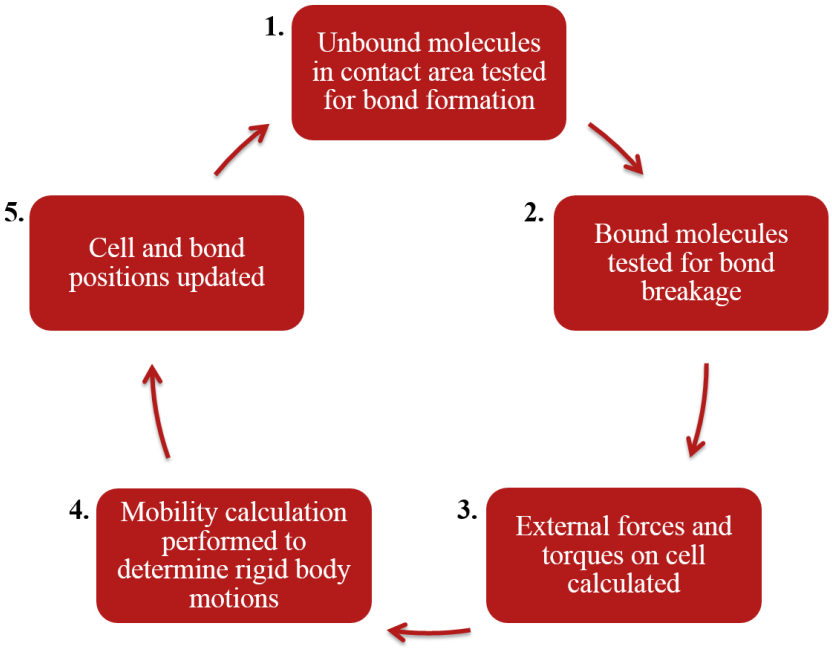

Figure 1.3: Multiparticle Adhesive Dynamics simulation solution algorithm. For a given iteration of the MAD simulation, 5 steps are performed. (1) Any unbound molecules on the particle in the contact area are tested for bond formation against the probability P_f. (2) Any existing bonds between the particle and the plane surface are tested for bond breakage against the probability P_b. (3) The external forces and torques on the cell, such as those from any existing bonds, fluid flow, and repulsive and gravitational forces, are calculated and summed. (4) The summed forces and torques are incorporated into the mobility calculation in order to determine the translational and rotational velocities of the particle. (5) From the results of the mobility calculation, the cell and bond positions are updated.

1.6 SUMMARY

Successful cancer metastasis hinges on the interaction between cells in circulation and selectins expressed on the endothelium. This metastatic process is highly complex, and it is a major factor contributing to the mortality of cancer. As a result, cancer cell adhesion to the endothelium, a critical step in the metastatic cascade, has been the subject of numerous *in vitro*, *in vivo*, and computational studies. Certain types of cancers, including breast, lung, liver, prostate, and colorectal, highly express selectin ligands and roll on activated endothelium in a manner similar to leukocytes. In comparison to solitary cells in circulation, circulating tumor cell clusters have an increased potential to survive, proliferate, and form secondary metastases, and the study of these clusters has been gaining interest. The use of numerical models to study CTM allows for insight to be gained on temporal and spatial scales not possible in physical experiments, and the MAD simulation can obtain unique information on the interactions between CTM and the endothelium. This model can successfully mimic the adhesive behaviors seen in *in vitro* experiments of tumor cell rolling, and data can be obtained for individual bonds as well as the behavior of the cell cluster as a whole. In my experiments, I analyzed model colorectal cancer cell line aggregates interacting with an E-selectin coated plane surface to compare the effects of aggregate size and confirmation on aggregate trajectory and adhesive interactions. By compartmentalizing the process of metastases and studying the effects of isolated receptor-ligand pairs that mediate rolling adhesion, the use of this model can potentially lead to the development of anti-metastatic treatments.

CHAPTER 2 – ADAPTATION OF MULTIPARTICLE ADHESIVE DYNAMICS SIMULATION TO MODEL CIRCULATING TUMOR MICROEMBOLI

2.1 INTRODUCTION

The method used by the Multiparticle Adhesive Dynamics (MAD) simulation to calculate rigid body motion, the Completed Double Layer-Boundary Integral Equation Method (CDL-BIEM), is one that can be applied to any arbitrarily shaped particle so long as its surface is continuous and relatively smooth, and it is well suited to model circulating tumor microemboli[103, 104]. This current adaptation, consisting of particles coated with sialyl Lewis^x (sLe^x) in linear shear flow in the presence of a single plane wall coated with E-selectin, serves as a simplified model for the initial phase of the metastatic adhesion cascade. Interest in endothelial selectins initially stemmed from the observation that complete inhibition of E- and P-selectin abolished leukocyte adhesion and migration[105], but E-selectin seems particularly vital to the tumor metastatic process. Sawada et. al.[106] found that the efficiency of colorectal cancer cell lines binding to both murine and human E-selectin was correlated with metastatic potential, and several studies have found that higher levels of soluble E-selectin expression was associated with increased metastasis and decreased prognosis in humans[107]. Prior studies have been undertaken to measure the binding mechanics of selectin tethers, but research has largely focused on selectin binding to P-selectin glycoprotein ligand-1 (PSGL-1). While selectins have a high level of sequence homology, they differ significantly in their binding kinetics and their ability to recruit leukocytes[108, 109]. As a major target for E-selectin binding, sLe^x has also been a popular target for study regarding tumor cell metastasis. Several studies have shown that higher expression levels of sLe^x in colon and other cancer cells are correlated with

metastasis and poor patient prognosis, and more malignant cancers tend to have higher sLe^x expression levels[21]. The use of a model system with linear shear flow and a flat plane serves as a representation of the flow conditions in linear portions of blood vessels, where the estimated range of blood flow forces are shear rates between 150-1600 s^{-1}, and resultant shear stresses between 0.8-4 dyne/$cm^{2,}$ depending on the vessel size and blood flow rate[30, 110].

In recent years, studies of colorectal, breast, and prostate cancer have served as a popular means to better understand the mechanisms underlying metastasis, and Colo205 cells are widely used when analyzing colorectal cancer cell behavior[22, 111-113]. Several kinetic parameters can be used to quantitatively describe the interaction between Colo205-bound sLe^x and E-selectin. These parameters – intrinsic on-rate of bond formation, intrinsic off-rate of bond dissociation, equilibrium bond length, bond reactive compliance, and bond tensile strength, dictate the rate that bonds form and break and the strength, length and duration of those bonds. Some of these parameters may be experimentally determined, but for other parameters there is currently no accurate way to measure them, so they must be obtained through model fitting. The intrinsic off-rate, k_r^o, which describes the probability of bond dissociation in the absence of applied force, can be measured using biomembrane force probe experiments, atomic force microscopy (AFM), or estimated using flow chamber experiments, but the intrinsic on-rate k_f^o cannot be accurately estimated, as the effects of cell collisions on the encounter rate cannot be isolated in currently existing flow assays[66, 114]. Estimations for the equilibrium bond length can be made based off X-ray

crystallographic structures of complexed ligands[115]. The bond reactive compliance, which describes the effective bond interaction range, can be calculated using various cell adhesion assays[57, 116], and although the bond tensile strength can also be estimated through adhesion assays, to the best of my knowledge those experiments have not been performed for this specific receptor-ligand pair. Estimates from similar selectin interactions serve as good approximations, as it has been found that rolling cell dynamics are not strongly dependent on the bond tensile strength[77, 78, 117]. Other key parameters, such as the magnitude and interaction range of the repulsive force, and the plane surface roughness layers, must be experimentally determined as well.

In order to adapt the model to describe Colo205/E-selectin interactions, I obtained all of the relevant parameters for the MAD simulation. I then validated the hydrodynamic behavior against analytical solutions for spherical aggregates in low Reynolds flow, also referred to as Stokes or creeping flow. There are widely available analytical solutions for the movement of rigid spheres in shear flow, both with and without the presences of a bounding wall, and derivations of that work have been developed for the movement of ellipsoids and some simple sphere aggregates. Much of the work in this field is based on the pivotal work of Jeffery, who described the motion of ellipsoids in creeping flow[118]. Nir & Acrivos expanded on that work and derived equations describing the motion of arbitrarily sized spherical doublets in creeping flow[119]. Other research has aimed at describing the behavior of chains of spheres or other aggregate conformations, and a modified equation for Stokes drag force has been derived to describe the effect of nonsphericity on their differences

relative to spheres[100, 120]. The agreement of the MAD simulation results with the computed analytical solutions would suggest it is an accurate model of the behavior of the created CTM.

2.2 METHODS

2.2.1 CELL CULTURE

The following was performed by King lab members Adelaide de Guillebon, Dr. Yue Geng, and Dr. Andrew Hughes. The human colorectal adenocarcinoma cell line Colo205 was obtained from American Type Culture Collection (ATCC number CCL-222) was obtained from ATCC (Manassas, VA) and maintained in Roswell Park Memorial Institute (RPMI) 1640 medium (Sigma) with 10% (v/v) fetal bovine serum (FBS) and 100 U/mL penicillin-streptomycin at 37°C in a 5% CO_2 incubator. Cells were imaged using an Olympus IX81 inverted microscope (Olympus America Inc., Melville, NY), linked to a television and a Sony DVD Recorder DVO-1000MD (Sony Electronics Inc., San Diego, California).

2.2.2 CELL RECEPTOR DENSITY CONJUGATION VIA FLUORESCENTLY CONJUGATED BEADS

The following was performed by King lab member Dr. Yue Geng. The Quantum™ FITC-5 MESF kit (Bangs Lab) including five microsphere populations (7-9μm) with assigned fluorescence intensity in MESF units was utilized to directly compare fluorescence measurements from Colo205 cells labeled with FITC-

conjugated anti-sLe^x monoclonal antibodies (mAb). The Quantum™ FITC-5 MESF kit was obtained from Bangs Lab (Fishers, IN). FITC-conjugated anti-sLe^x mAb (CD15s) were purchased from Santa Cruz Biotechnology (Dallas, TX). QuickCal® was used in determining the expression level of cells, and for evaluating instrument linearity and detection threshold. Briefly, Colo205 cells were washed and re-suspended in 1X Dulbecco's phosphate-buffered saline (DPBS) with 1% bovine serum albumin (BSA) to a final concentration of 200,000 – 300,000 cells in each sample. sLe^x mAb and appropriate isotype control were added to cell suspensions and incubated on ice for 45 min. Following the incubations, the cells were washed three times with 1 mL of 1X DPBS to remove any unbound antibody. Flow cytometry samples were analyzed using an Accuri C6 flow cytometer (Accuri Cytometers Inc., Ann Arbor, Michigan) and plots were created using the FCS Express package. The Colo205 fluorescence was then directly compared with the fluorescence of polymorphonuclear neutrophils (PMNs), of which the sLe^x surface expression has previously been quantified[68]. Isolation of PMNs has been described previously[80]. Averaged values were reported as mean ± standard deviation (SD).

2.2.3 ELECTRON MICROSCOPY OF CELL SURFACE

The following was performed by King lab member Dr. Andrew Hughes. Colo205 cells were released from the culture flask using Accutase (MP Biomedicals, Solon, OH), washed once with PBS (Invitrogen, Camarillo, CA), and placed on ice. Cells were fixed for 2 h in 2% glutaraldehyde (Electron Microscopy Sciences,

Hatfield, PA) on ice. Cells were subsequently treated with 1% osmium tetroxide (Electron Microscopy Sciences) for 1 h on ice. Fixed cells were dehydrated with successive 5 min washes of 25% and 50% ethanol. Cells were allowed to sit in 75% ethanol overnight at 4°C and then washed twice for 5 min in 100% ethanol. Cells were subjected to critical point drying with carbon dioxide, mounted on metal stubs with copper tape, and coated with carbon. These instruments were provided by the Cornell Center for Materials Research. Samples were stored in argon at room temperature and then imaged with a Zeiss Ultra scanning electron microscope (SEM) at magnifications of X6,000 to X30,000 in the Cornell Nanofabrication Facility.

2.2.4 QUANTIFICATION OF COLO205 MORPHOLOGY

I analyzed optical light and scanning electron microscopy images of cultured Colo205 cells and cell clusters using ImageJ, an open source image processing software available at [https://imagej.nih.gov/ij/index.html]. For the optical microscope images, the major and minor axes of 85 cells were measured using the straight line tool, and those values were averaged to determine mean cell diameter. The cell surface roughness layer was determined though ImageJ analysis of the electron microscopy images. In this model cells are assumed spherical, so surface roughness elements were defined as bumpy features that extended beyond the spherical shape of the cell. In ImageJ the spherical boundary was traced, and then paired points were recorded from the boundary of the cell to the height of surface features to determine their heights. I measured a total of 181 surface roughness elements, and the surface roughness layer

length was defined as the mean surface roughness feature height. An example of the analysis is shown in Fig. 2.1. Averaged values were reported as mean ± SD.

2.2.5 QUANTIFICATION OF E-SELECTIN LIGAND DENSITY

The following was performed by King lab member Dr. Andrew Hughes. Polyurethane microtubules (0.012" inner diameter), purchased from Braintree Scientific (Braintree, MA), were incubated with a 10 µg/mL solution of protein G Calbiochem (Gibbstown, USA). In 1x DPBS for 1 h and 3,5, or 15 µg/mL human E-selectin/CD62E Fc chimera purchased from R&D systems (Minneapolis, MN) in DPBS for 2h. E-selectin concentration was determined by comparing the concentration of E-selectin in solution before and after incubation. Protein concentration in solution was determined by modified Lowry assay using the DC™ Protein Assay (Bio-Rad, Hercules, CA).

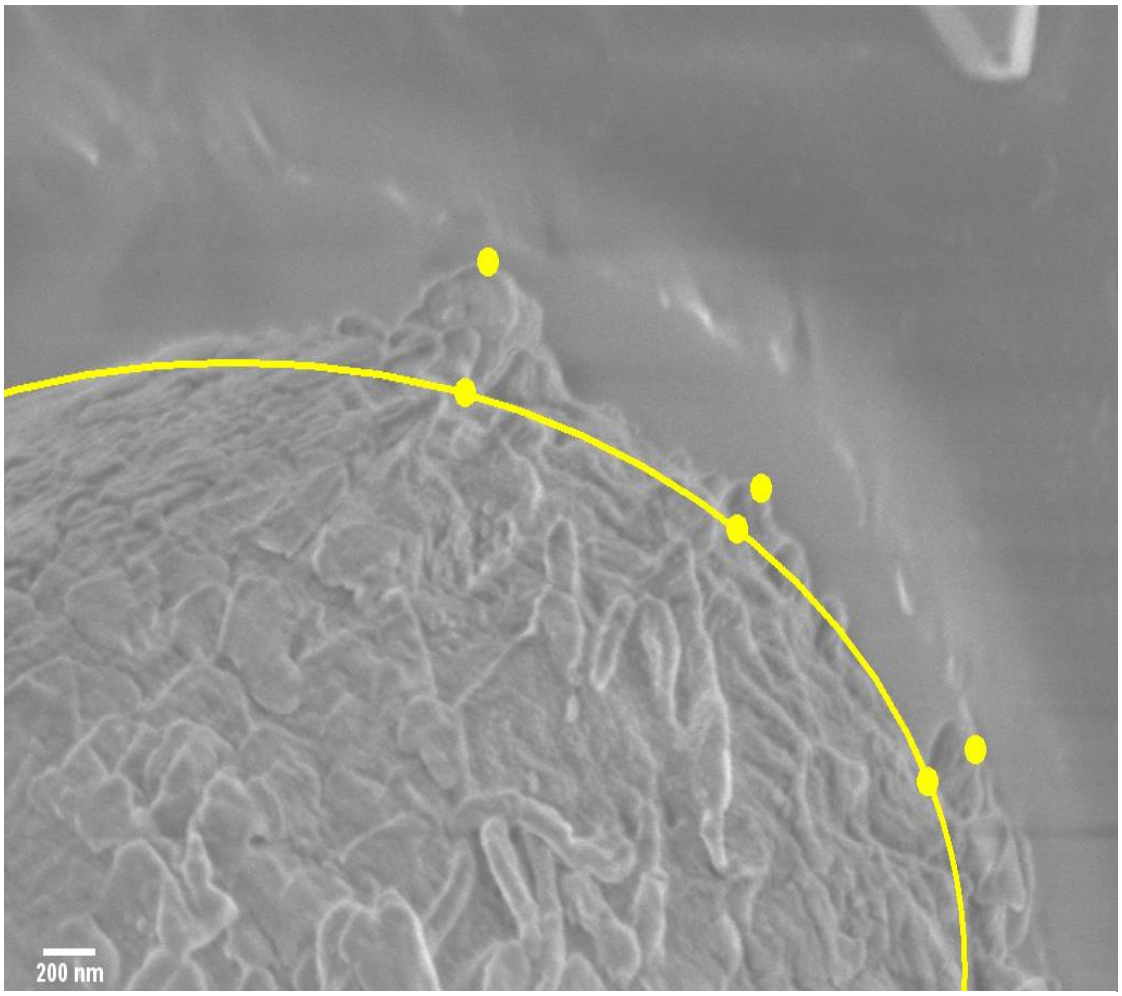

Figure 2.1: Analysis of scanning electron microscope images of Colo205 cells. Representative scanning electron microscope image of a Colo205 cell with surface roughness elements labeled for calculation of mean surface roughness. The oval selection tool in ImageJ was used to outline the edge of the imaged cells, and the multi-point selection tool was then used to mark the base and peak of surface roughness elements. The distance between the paired points was measured to determine the mean height of the roughness features extending from the surface of the cell.

2.2.6 GENERATION OF MULTIPARTICLE ADHESIVE DYNAMICS SIMULATION MODEL PARTICLES

I developed an algorithm using MATLAB software (The MathWorks Inc., Natick, MA) to modify the original sphere geometry used in the MAD simulation. Model particles in the original iteration of the MAD simulation where spheres discretized into QUAD9 elements, quadrilateral elements with three nodes per side and one central node, for a total of 9 nodes per element. The most recent iteration of the MAD simulation used 96 element spherical particles. These are generated by starting with a cube, dividing each cube face into 16 quadrilateral elements, and then projecting the cube onto the surface of a sphere placed at the cube center[91, 121]. I developed mesh geometries using the 96 QUAD9 element sphere as a template, and modified them using morphological data collected from Colo205 cell aggregate samples. I created mesh geometries for 2-4 cell aggregates in idealized conformations of those observed in *in vitro* studies. To create a doublet mesh, the developed algorithm removes one face of the sphere and then creates a duplicate of it. The duplicated sphere is then reoriented so that the opened faces are aligned towards each other and the two spheres are in contact with each other. The nodes along the open face of each sphere are then merged so that the mesh consists of one continuous set of elements. This algorithm allows any number of faces to be removed from the original sphere, and a corresponding sphere duplicate will be attached to that side. Attachments can also be made to any duplicated spheres in the aggregate, rather than the original. The coordinates for each monomer can be adjusted in tandem or separately to create

CTM that match the morphology of imaged Colo205 cells. To demonstrate the effectiveness of the mesh generation, I created CTM doublet, triplet, and 4-mer mesh geometries to match aggregates observed *in vitro*. For the experiments, idealized aggregates were used. I generated mesh input files for CTM doublet, triplet, and 4-mer of equally sized spheres, with multiple conformations for triplet and 4-mer aggregates.

2.2.6 NUMERICAL IMPLEMENTATION

I modified the Multiparticle Adhesive Dynamics model, developed by King and Hammer[91, 92], from the configuration optimized for leukocyte/P-selectin interactions in order to describe the interaction between Colo205-bound sLe^x and E-selectin. The solution algorithm for the model is as follows:

1. If the cell is within reactive distance, all unbound molecules in the contact area are tested for bond formation against the probability P_f' (Equation 2.2)
2. All currently bound molecules are tested for bond breakage against the probability P_r (Equation 1.11)
3. The external forces and torques on each cell are calculated by summing over adhesive forces and adding nonspecific repulsive and gravitational forces
4. The mobility calculation is performed to determine the rigid body motions of the cells
5. Cell and bond positions are updated according to the kinematics of cell motion

The MAD simulation code was written in FORTRAN 95 and compiled with the GNU Fortran Compiler; double precision was used for all calculations. I ran computations

using Dell R420 16-core, Dell R410 12-core, and SunFire quad-core processor servers. Runtime was a function of mesh size and the number of iterations required for CDL-BIEM results to converge at each timestep, and varied largely between groups. Simulation durations ranged from a few days to a few weeks. A dynamic timestep was incorporated into all simulations; the timestep reduced during bond events and as the number of iterations required for convergence increased, which typically coincided with decreased cell to surface distance. For purely hydrodynamic simulations, the minimum timestep was set to 10^{-7} sec, and for simulations with adhesive interactions, the minimum timestep was set to 10^{-6} sec. Equation 1.8 was employed for the repulsive force calculations, with experimentally determined values for the magnitude of repulsive force F_o and repulsive force interaction range τ.

For the best model fit, simulation parameters were extrapolated from experiments done for this specific ligand-receptor pair when possible. The size, sLe^x receptor density, and surface roughness were measured for cultured Colo205 cells and incorporated as input parameters for the MAD simulation. I used information from the *in vitro* experiments and the literature to determine the kinetic parameters such as intrinsic off-rate and bond reactive compliance. The unknown parameters had to be determined through model fitting. To determine the repulsive force parameters, I initially ran simulations using only the hydrodynamic component of the MAD simulation. The magnitude or interaction range of the repulsive force was held constant, and the other variable were altered over a range of values in order to achieve a repulsive force strong enough to repel the particles from the surface. The initial

values were $F_o = 5.0*10^8$, $\tau = 2000\ \mu m^{-1}$, from the previous iteration of the MAD simulation. Once a parameter combination was found capable of repelling particles from the surface, I enabled the Monte Carlo component of the MAD simulation, and the parameters were systematically altered again to ensure that the repulsive force was strong enough to repel a particle that had formed bonds with the surface. To determine the bonding parameters, I ran several iterations of the MAD simulation with multiple permutations of a range of values for each unknown parameter, holding all other parameters fixed, monitoring the particle velocity, instantaneous bond number, bond lifespan, and bond length. The simulation results were compared with the results of theoretical solutions of particle motion in flow and in-house parallel plate flow chamber assays to determine the optimum values for each parameter.

Additions to the model were also made to increase model fit and efficiency. An implication of the bond formation probability previously described is that the effect of ligand density on the plane surface is not directly incorporated into the equation. However, selectin-mediated rolling velocity is highly dependent on E-selectin density[63]. To account for specific surface densities, I added a parameter containing the ratio of the molecule density on the plane surface to that of the cell surface:

$$ligand\ ratio = \frac{density\ of\ E{-}selectin\ on\ plane\ surface}{density\ of\ sLe^x\ on\ particle\ surface} \tag{2.1}$$

The ligand ratio was included in Equation 1.9 to form a modified relation for bond formation:

$$P_f' = \frac{P_f}{ligand\ ratio} = 1 - \exp(-k_f \Delta t) \tag{2.2}$$

When testing the probability of bond formation, the association rate k_f is calculated for a given molecule, and a probability of formation $P_f^{'}$ is determined. A random number is then generated, and if that random number is less than $P_f^{'}$, a bond forms during that timestep. This modified bond formation relation results in E-selectin density directly affecting the probability of bond formation. The density of E-selectin used in the model was the same as the density of E-selectin used in the *in vitro* rolling experiments. I also adapted the Monte Carlo component of the MAD simulation to run via parallel processing using the OpenMP application program interface in order to decrease simulation run times.

To validate the hydrodynamic behavior of the CTM, I placed particles at an initial centroid height 30 volume-equivalent sphere radii from wall, with major axes oriented perpendicular to flow, and any cell protrusions oriented in the positive axis direction. At this distance, wall effects could be safely neglected[122]. The shear rate was set to 1000 s^{-1}, and the CTM were allowed to translate and rotate in fluid flow for a simulated time of 1 s. For the doublet, the rotation rate Ω vs. orientation angle θ was compared to the general expression given by Nir & Acrivos for [119]

$$\Omega = \frac{d\theta}{dt} = \frac{1}{2}\, G\, (1 + C\cos 2\theta) \tag{2.3}$$

Where G is the shear rate, and C is a constant defined as

$$C = \frac{r_e^2 - 1}{r_e^2 + 1} \tag{2.4}$$

Here, r_e is the ellipsoid equivalent axis ratio. The equivalent axis ratio for a rod-like chain of spheres is determined using the semi-empirical relation given by Harris and

Pitman[123]:

$$r_e = 1.14a_r^{0.844} \tag{2.5}$$

Where a_r is the true aspect ratio of any axisymmetrical particle, defined here as (major axis length/minor axis length). The aspect ratio of the doublet was 1.929, resulting in a C value of 0.576. For the triplet and 4-mer rods, the rotation rate vs. orientation angle was compared to Jeffery's equation for the rotation rate of an ellipsoid in unbounded shear flow[118]:

$$\frac{d\theta}{dt} = \frac{G}{r_e^2+1}(r_e^2 sin^2\theta + cos^2\theta) \tag{2.6}$$

The aspect ratios for the triplet and 4-mer were 2.859 and 3.788, respectively. All rotation rates were normalized by the shear flow rate. I calculated the normalized root-mean-square deviation (NRMSD) for each rotation rate to determine the model error. The root-mean-square deviation (RMSD) was defined as

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(x_i' - x_i)^2}{n}} \tag{2.7}$$

Where n is the number of data points, x_i is the observed value at i, and x_i' is the theoretical value at i. The RMSD was normalized by

$$NRMSD = \frac{RMSD}{x_{i,max} - x_{i,min}} \tag{2.8}$$

I compared the Stokes drag force for aggregate conformations with analytical solutions available. The drag force on a sphere in low Reynolds shear flow is given by

$$F_D = 3\pi\mu Ud \tag{2.9}$$

where μ is the fluid viscosity, U is the fluid velocity relative to the sphere, and d is the

sphere diameter. For non-spherical particles, the drag force must be modified:

$$F_D = 3\pi\mu U d_e K \tag{2.10}$$

The sphere diameter must be replaced with the volume-equivalent sphere diameter d_e

$$d_e = \left(\frac{6}{\pi} Volume\right)^{1/3} \tag{2.11}$$

and a shape factor K must be added. As an object's nonsphericity increases, the surrounding flow field becomes increasingly dissimilar to that of a sphere, causing the drag to be higher than predicted using the standard Stokes flow equation[124]. The shape factor has been calculated for simple conformations, and those values are given in Table 2.1[120]. I also determined the relative error for the drag force calculations.

Table 2.1: Stokes drag force equation shape factor values.

Aggregate Shape	Shape Factor
OO	K = 1.12
OOO	K = 1.27
O O O	K = 1.16
OOOO	K = 1.32
OO OO	K = 1.17

2.3 RESULTS

2.3.1 ESTABLISHMENT OF MODEL CTM GEOMETRIES

To demonstrate the capability of the mesh generation algorithm to create relevant model particles, I created doublet, triplet, and 4-mer particle meshes to match the morphology of imaged Colo205 CTM, shown in Fig. 2.2. Accurate representations of imaged CTM could be made by increasing or decreasing monomers relative to the average size, and adjusting the coordinates at the intersections between monomers. For subsequent experiments, CTM displayed in in Fig. 2.3 were used; all aggregate monomers had a radius of 6.959 μm, and connections between monomers occurred through the central 4 QUAD9 elements of each sphere face. A 184 element doublet mesh was created for the one conformation of two equally sized cells in contact. For the triplet, two conformations were created. A 272 element triplet arranged as a rod-like chain of three cells, and a 272 element triplet in a triangular formation. For the 4-mer, three conformations were created: a 360 element rod-like chain of spheres, a 352 element square, and a 360 element tetrahedron.

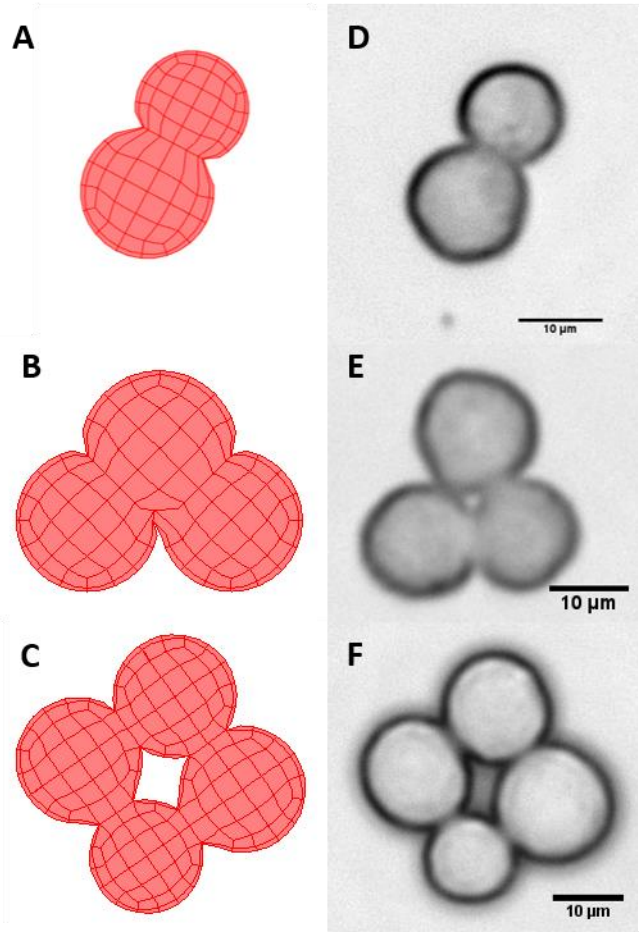

Figure 2.2: QUAD9 element models of Colo205 aggregates. QUAD9 element meshes were generated for doublet, triplet, and 4-mer aggregates using a MATLAB algorithm developed to create particle geometries using a 96 element sphere mesh as a template. Aggregates were created as clusters of spheres of equal size, and then the nodal coordinates were adjusted so that the model particles (A-C) matched the appearance of cultured Colo205 aggregates (D-F) imaged using an inverted microscope.

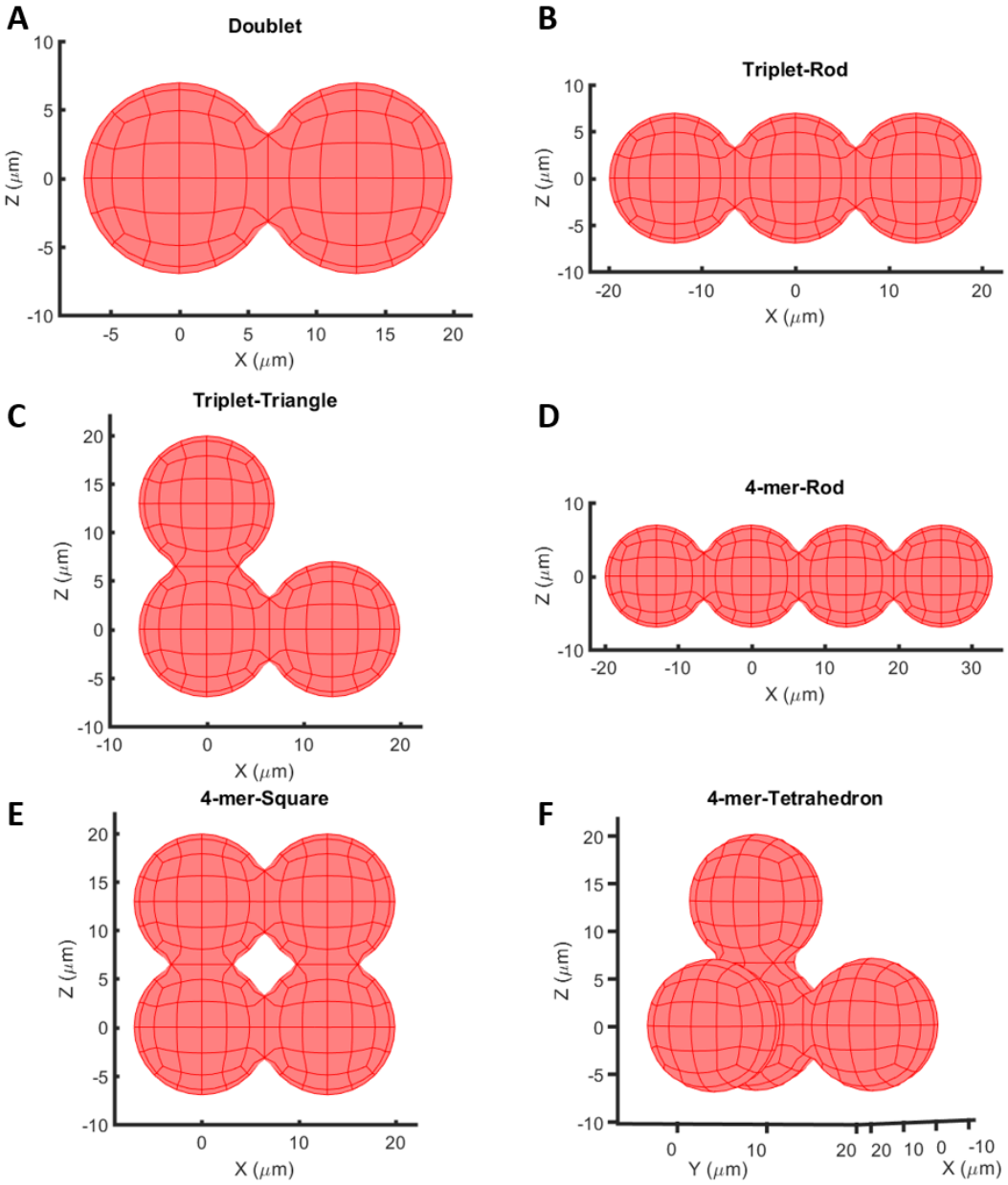

Figure 2.3: QUAD9 element sphere aggregates. Model aggregates of equal sized spheres were created for simulations of CTM behavior in Stokes flow. Aggregates consisted of clusters of 2, 3, or 4 cells in common geometrical configurations. Particles in (A-F) contained 184, 272, 272, 360, 352, and 360 QUAD9 elements, respectively.

2.3.2 QUANTIFICATION OF SIALYL LEWIS X DENSITY ON COLO205 CELLS

The Quantum™ FITC-5 MESF (Molecules of Equivalent Soluble Fluorochrome) kit was used to establish a calibration curve relating flow cytometer channel values to standardized fluorescence intensity (MESF) units. With the calibrated flow cytometer, the MESF was compared for Colo205 cells and PMNs incubated with 10, 20, and 50 μg/mL anti-sLe^x mAb, shown in Fig. 2.4. The MESF of Colo205 cells at all concentrations was 3.24 ± 0.506 (SD) times higher than the MESF of PMNs. The use of the Quantum™ FITC-5 MESF kit ensured the fluorescence was directly proportional to sLe^x expression. Given the density of sLe^x on PMNs, previously calculated by Rodgers et al.[68], the density sLe^x of Colo205 cells was determined to be 1341 molecules/μm^2.

2.3.3 MAD SIMULATION PARAMETERS

Table 2.2 lists the input parameters for the MAD simulation. Cell radius, sLe^x surface density, E-selectin surface density, and surface roughness layer height were obtained from the aforementioned characterization studies. The mean cell diameter measured was 13.92 ± 2.42 μm. E-selectin substrate densities were calculated for incubation concentrations of 3, 5, and 15 μg/mL E-selectin. They were 1575.5, 2625.9, and 7877.6 molecules/μm^2, respectively. For flow assay experiments in these studies, an incubation concentration of 5 μg/mL E-selectin was used, so 2626 molecules/μm^2 was used as E-selectin density in the simulation. Mean surface roughness feature height was 270.6 ± 148.1 μm.

The bond tensile strength, intrinsic on-rate, and magnitude and interaction range of repulsive force were determined through model fitting. For parameter values obtained from the literature, the relevant study was cited.

2.3.4 VALIDATION OF SIMULATION BEHAVIOR

The rotational rate of rod-like doublet, triplet, and 4-mer aggregates placed far from the plane surface was compared to theoretical solutions for the rotation of rod-like particles in unbounded shear flow. Fig. 2.5 show that the results were in excellent agreement. The relative error for the rod-like doublet, triplet, and 4-mer was 2.68, 0.88, and 1.63%, respectively. At $\theta = 0$, the particles are oriented with their major axis parallel to flow, and as can be seen from the graph, the largest angular velocity occurred when the particles were oriented perpendicular to flow. The particles exhibited very similar periodicities of rotation, and the small difference from theory can be attributed to discretization error.

The Stokes drag force of model aggregates placed far from the plane surface was compared to theoretical solutions for the Stokes drag force of simple sphere clusters, shown in Fig. 2.6. These results are also in excellent agreement; the relative errors were 0.006, $6.3*10^{-4}$, 0.26, 0.001, and 0.009% for the doublet, triplet-triangle, triplet-rod, 4-mer-square, and 4-mer-rod, respectively.

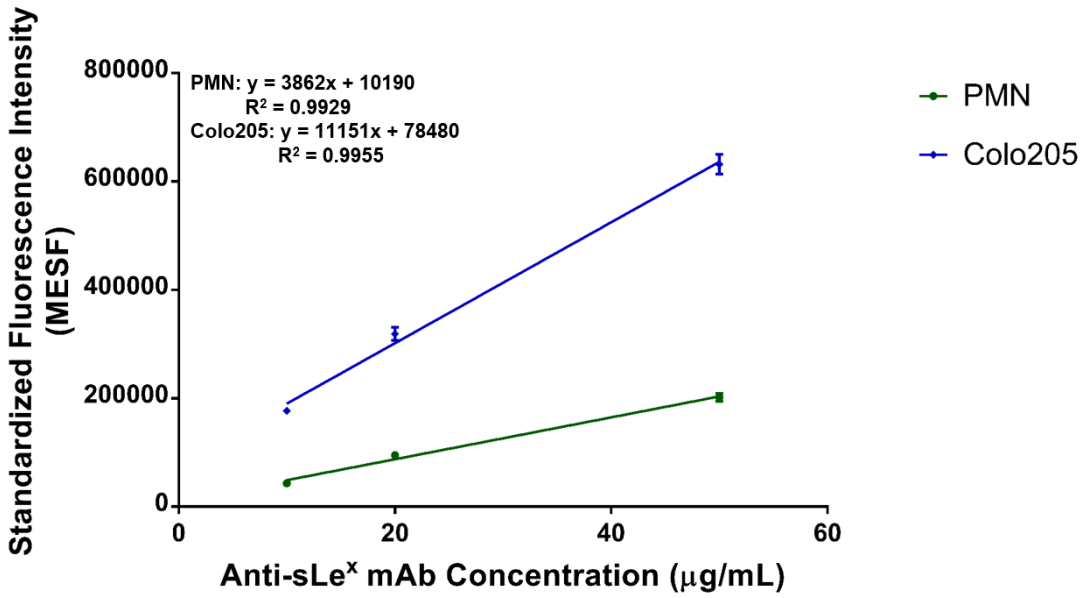

Figure 2.4: Surface expression of sialyl lewis^x on polymorphonuclear neutrophils and Colo205 cells. Mean fluorescence intensity of polymorphonuclear neutrophils (PMNs) and Colo205 cells stained for sialyl lewis^x (sLe^x). The flow cytometer was calibrated using the Quantum™ FITC-5 MESF kit so that the fluorescence measurements were directly proportional to expression of sLe^x. Fluorescence intensity, and therefore sLe^x expression, was on average 3.24 ± 0.506 (SD) times higher in Colo205 cells. For symbols with no visible error bars, error bars were smaller than the symbol marker.

Table 2.2: Values of input parameters used in Multiparticle Adhesive Dynamics simulation. Key input parameters used in the MAD simulation. Values relating to cellular features were specifically measured for Colo205 cells, and values relating to receptor-ligand bonding were specifically collected for the interaction between sialyl Lewis^x (sLe^x) and E-selectin. Parameters highlighted in grey were measured through *in vitro* experiments. Cell radius was determined through measuring the diameters of cultured Colo205 cells imaged with an inverted microscope, assuming cells were spherical. Surface density of sLe^x was determined through quantitative flow cytometry analysis. Surface density of E-selectin was determined by comparing the concentration of E-selectin in solution before and after incubation of flow device in preparation for *in vitro* flow assays. Room temperature was used for *in vitro* flow assays and MAD simulation experiments. Particle surface roughness was determined through measuring the height of surface roughness features imaged with a scanning electron microscope. Parameters highlighted in red were determined through model fitting. Non-highlighted parameters were obtained from relevant *in vitro* studies, with a reference to the particular study listed.

Parameter	Definition	Value	Reference
R	Cell radius	6.9590 μm	
m_r	sLe^x surface density	1341 molecules/μm^2	
m_l	E-selectin surface density	2626 molecules/μm^2	
λ	Equilibrium bond length	20 nm	[77]
σ	Bond tensile strength	100 dyn/cm	
γ	Bond reactive compliance	0.2 Å	[116]
k_f^o	Intrinsic on-rate	4.0 s^{-1}	
k_r^o	Intrinsic off-rate	0.44 s^{-1}	[114]
F_o	Magnitude of repulsive force	2.4*10^9 pN	
τ	Repulsive force interaction range	1666.667 μm^{-1}	
μ	Fluid viscosity	1.0 cP	[91]
ρ_f	Fluid density	1.0*10^{-6} μg/μm^3	[91]
ρ_p	Particle density	1.05*10^{-6} μg/μm^3	[91]
T	Temperature	298 K	
ε_w	Wall surface roughness	350 nm	[125]
ε_p	Particle surface roughness	270.6 nm	

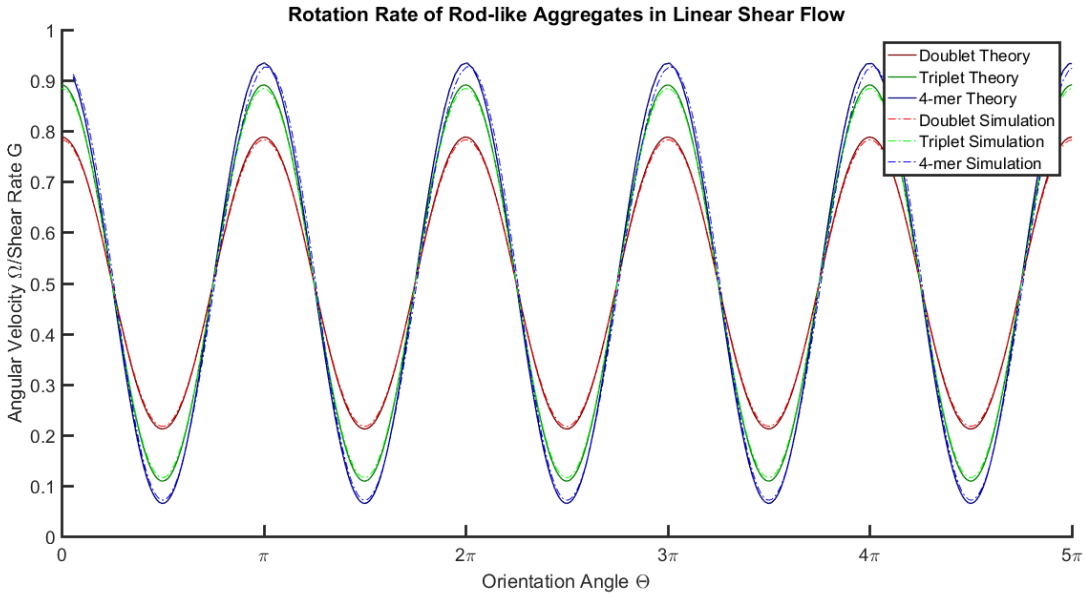

Figure 2.5: Rotation rate of model aggregates matches analytical solutions. The normalized angular velocity vs. orientation angle of rod-like aggregates placed far from a bounding wall was found to be in excellent agreement with solutions given by Equations 2.2 and 2.5. The maximum rotation occurred when the aggregates are aligned perpendicular to flow ($\theta = 0, \pi, 2\pi, 3\pi, ...$), and the minimum occurred when aggregates are aligned parallel to flow ($\theta = \frac{\pi}{2}, \frac{3\pi}{2}, \frac{5\pi}{2}, \frac{7\pi}{2}, ...$).

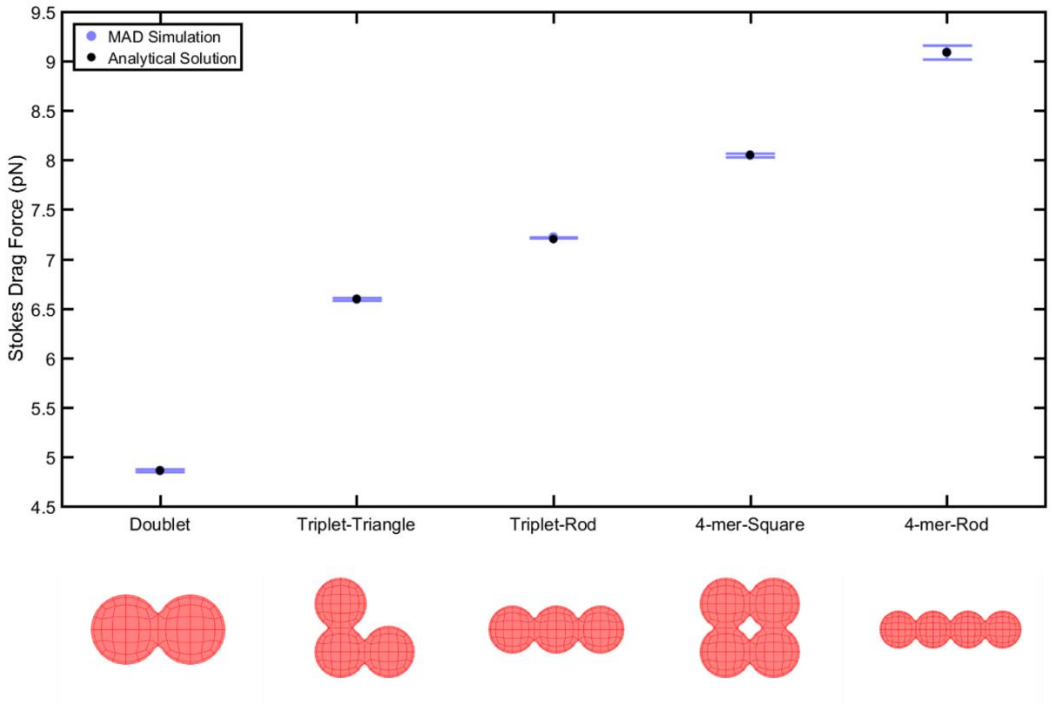

Figure 2.6: Stokes drag force of aggregates matches analytical solutions. The Stokes drag force of aggregates placed far from a bounding wall was found to be in excellent agreement with analytical solutions given by Equation 2.10. The relative error for all aggregates was less than 1%.

2.4 DISCUSSION

The simulation parameters extrapolated from the characterization experiments complement the current understanding of CTC morphology. As expected for the measurements of Colo205 cells, there was a large heterogeneity in cell size, but the average follows the well-acknowledged trend of CTCs having a larger size than other blood cells[126, 127]. The CDL-BIEM calculations for motion assume that the particles are rigid spheres; this is a reasonable assumption for the Colo205 cells. While some cells, such as breast cancer, are soft and more deformable, both leukocytes and colorectal CTCs are nearly spherical and, prior to firm adhesion, rigid cells with limited deformation[127-131]. The surface expression of sLe^x for Colo205 was 3.24 times greater than the expression levels of PMNs, which is reasonable considering human colon cancer tumors are known for overexpression of sLe^x-containing mucins[21]. The value calculated for the particle surface roughness, 270.6 nm was within the acceptable range for the assumption of unperturbed flow near the surface of the particle. The fluid force and fluid moment acting on a cell is insensitive to the value of surface projections when they are less than 0.12 r, which for a model Colo205 cell would be 0.835 μm[132]. The parameters obtained through model fitting, particularly the bond tensile strength and repulsive force parameters, are comparable to those used in prior iterations of the adhesive dynamics simulation[91, 92, 121]. The value for the plane surface roughness is based off the thickness of the endothelial glycocalyx, which has been estimated to have a thickness of 0.3-0.5 μm[125].

The agreement of the model aggregates with analytical solutions supports the utility of the MAD simulation in predicting the behavior of particles in flow. The fit of the rod-like particles to analytical solutions for Stokes drag force and angular velocity were to be expected, as the behavior of rod-like particles and most other axisymmetric bodies follows Jeffery's orbit analysis[133]. The discrepancies in behavior can be attributed to discretization error, which could be reduced with the use of a more refined mesh, at the cost of computation time. Also, the equation for the ellipsoid equivalent ratio of rod-like particles was originally developed for cylindrical rods, which have more blunted ends than a chain of spheres[100]. The model CTM drag forces fit with analytical solutions with less than 1% error, though the triangular triplet had relatively higher errors than the other groups. This can be attributed to the shape factor, which was based off a triangular aggregate where each monomer is in contact with the other two, whereas the monomers in the model triplet only form contact with one other monomer. However, the agreement of particle drag force with theory suggests that this deviation in conformation is minor.

2.5 CONCLUSIONS

Colo205 cell morphology and surface receptor expression were successfully characterized, and relevant physiological parameters were extrapolated for use in the MAD simulation. The input parameters established a model system optimized for the analysis of E-selectin/sLe^x interactions. The particle parameters used, cell radius, sLe^x expression, and surface roughness, were all specifically measured for Colo205 cells.

The parameters related to receptor-ligand binding, intrinsic on-rate and off-rate, bond tensile strength, bond reactive compliance, and equilibrium bond rate, and plane surface ligand density, were specifically collected for the E-selectin/sLe^x receptor-ligand pair. An algorithm was developed capable of generating particle meshes for a wide variety of cell aggregates, including doublets, triplets, and 4-mer aggregates in rod-like and more spherical conformations. The size and shape of the entire aggregate or portions of it could be altered to match aggregates found *in vitro*, but for subsequent experiments I created idealized aggregates containing spherical monomers of equal size. The model showed an excellent fit with analytical solutions for flow behavior in low Reynolds shear flow, suggesting the utility of this model as a predictor for the behavior of CTM as they make initial contact with activated endothelium.

CHAPTER 3 – HYDRODYNAMIC BEHAVIOR OF CIRCULATING TUMOR MICROEMBOLI

3.1 INTRODUCTION

The initial interaction between a circulating tumor cell (CTC) and activated endothelium is a crucial step in the metastatic process that can only occur given the proper balance of forces. The formation of a bond between two molecules requires two major steps: transport, where the molecules are brought into close proximity, and reaction, where the molecules link together. Many biochemical reactions, such as the cytokine activation of endothelial cells, involve molecules in solution forming complexes after diffusion driven collisions with cell-bound ligands. In contrast, with selectin-mediated bonding a moving cell carries adhesion receptors to other cell-bound ligands, thus transport is driven by the relative velocity of involved cells. In steady laminar flow, the velocity of the fluid, and that of flowing particles, increases with the distance from the wall, and this effect is highly dependent on the size and shape of the given particle. At a given distance, larger cells will have higher velocities, as increasing portions of the cell will extend further from the wall. A faster cell velocity means that adhesion molecules will be brought into contact more frequently, but as the velocity increases the contact time is decreased. Thus, optimum binding efficiency depends on the relative timescales of transport and binding reaction[23, 102]. This is particularly important with selectin binding. Numerical models of leukocyte binding to endothelium show that reaction rate increases with the relative velocity of the interacting molecules and then reaches a plateau, when the high encounter rate and low duration counterbalance each other[102].

There has been extensive study of the motion of spherical particles in shear

flow, both with and without the presence of a bounding wall, but the study of spherical aggregates is more difficult, as analytical solutions are unavailable for irregular geometries. Studies have analyzed the behaviors of nonspherical particles, and those have highlighted the important effects of nonsphericity on transport and hydrodynamic interactions, particularly when in close proximity to a wall. Far from a wall, shear force induces a sphere to translate in the direction of flow with a constant rotational velocity. The presence of a wall creates a velocity component normal to the wall, and as a result the sphere will undergo an oscillatory motion towards and away from the wall. This wall effect only occurs over a short distance, and may be considered negligible after a distance of 5-10 minor axes. The wall induces a drag on the sphere, slowing its rotation relative to the fluid. The wall effect induces different behavior in nonspherical particles. Studies of ellipsoids and rod-like particles show that rotation is dependent on the particle orientation, reaching a maximum when the particle is oriented perpendicular to flow. When these particles are close to the wall, they rotate with a tendency for the major axis to align parallel to flow, resulting in a larger rotation period relative to unbounded flow. This period increases with higher aspect ratio, as the particle spends more time aligning parallel to flow. The presence of a wall affects a nonspherical particle differently depending on its aspect ratio and orientation. When the particle is oriented with its major axis more perpendicular to flow, the wall reduces the fluid motion along the particle's surface in the z-direction. This reduces the viscous friction on the upper edge of the particle, reducing its torque relative to unbounded flow. When the particle is oriented with its major axis more parallel to

flow, the wall reduces the fluid motion in the gap between the particle and the wall, resulting in lower viscous friction only on the side of the particle facing the wall, and thus increased torque. This wall effect on torque is not present for spheres, and it increases with aspect ratio. The presence of a wall also imposes hydrodynamic interactions on nonspherical particles that are not present with spherical particles. Unlike with spheres, the presence of a wall imposes a lift force on particles, and this effect is most pronounced for particles with intermediate aspect ratios. Spherical particles experience no lift, and with particles with high aspect ratios, major portions extend away from the wall, where wall-particle interactions are negligible[98, 100, 122, 134]. These wall-particle interactions alter the trajectories of particles that come in proximity, and such changes can affect the capability of flowing cells to form adhesive interactions. *In vivo*, contact with the endothelium is largely promoted through collisions with blood cells, but *in vitro* assays aimed at understating the behavior of cells in flow utilize dilute suspensions in order to isolate more subtle mechanisms, and in those environments understanding hydrodynamic interactions is vital[92].

Investigating the hydrodynamic behavior more of complicated geometries must be done with numerical models, with inferences drawn from our current understanding of related geometries. The potential for circulating tumor microemboli (CTM) to interact with endothelium can be investigated using the hydrodynamic component of the Multiparticle Adhesive Dynamics (MAD) simulation. To that end, I ran two series of simulations to analyze the hydrodynamic behavior of model CTM at various heights, shear rates, and orientations near a plane wall, to observe the

trajectories of CTM and estimate their potential for adhesive contact.

3.2 METHODS

3.2.1 COLLISION PARAMETERS EVALUATED

Model CTM geometries, displayed in Fig. 2.3, were systematically tested under physiologically relevant flow conditions for collision events. A model particle was considered to have undergone a collision event if any node of the particle came within reactive binding distance of the plane surface. Reactive binding distance was defined as surface roughness layers being at a distance less than or equal to the equilibrium length of an E-selectin/sLe^x bond, 20 nm. Collision contact time and contact area were recorded for each collision, and time integral of contact area was recorded for each model CTM particle. Time integral of contact area (TICA) is a measure of relative adhesion probability[93], defined as

$$\int_{t_i}^{t_i+t_c} A_c dt \tag{3.1}$$

where t_i is the initial time of collision, t_c is the contact time during collision, and A_c is the contact area, or the surface area of the particle within reactive distance to the plane wall. Simulation times were normalized by the shear rate, and distances were normalized by the volume-equivalent sphere radius R_e. Table 3.1 lists the R_e values for each model CTM.

Table 3.1: Relevant physical characteristics of model circulating tumor microemboli. Volume-equivalent sphere radius, aspect ratio, and ellipsoid-equivalent axis ratio are common metrics used to characterize nonspherical particles. Aspect ratio is defined as the ratio between the longest and shortest particle axis. The ellipsoid-equivalent axis ratio is defined as the aspect ratio of an ellipsoid of the same volume. For a chain of spheres, $r_e = 1.14 a_r^{0.844}$, where a_r is the true aspect ratio. The ellipsoid equivalent axis ratio was not calculated for the triangular triplet, or 4-mer rod, square, or tetrahedron, as those conformations differed too greatly from an ellipsoid.

Model Particle	Volume-equivalent Sphere Radius R_e (μm)	Aspect Ratio	Ellipsoid-equivalent Axis Ratio r_e
Doublet 	8.762	1.929	1.929
Triplet-Triangle 	10.028	1.929	
Triplet-Rod 	10.028	2.859	2.881
4-mer-Rod 	11.036	3.788	3.673
4-mer-Square 	11.033	1.929	
4-mer-Tetrahedron 	11.054	1	

3.2.2 NUMERICAL IMPLEMENTATION

A series of simulations were run with model particles of each conformation listed in Fig. 2.3 were placed with an initial centroid height of 0.25 volume-equivalent sphere radius R_e from the surface, with major axes oriented parallel to flow, with minor axes centered at the origin, and allowed to translate and rotate freely in shear rates of 500, 1000, and 2000 s^{-1} for a simulated time of 1 s. As the hydrodynamic component of the MAD simulation is deterministic, no duplicate simulations were necessary. A second series of simulations was run with model particles placed at an initial centroid height of 0.25, 0.5, 0.75 and 1 R_e with arbitrary orientations, and allowed to translate and rotate freely at a shear rate of 1000 s^{-1}. In addition to recording the contact time, contact area, and TICA for each collision, the simulation recorded instantaneous position, orientation angle, and time every 100 time steps.

I developed MATLAB algorithms to visualize the simulations. One algorithm created produced a video of the trajectory of the particle over time, with each video frame containing a multiple views of the particle simultaneously. The side, back, and bottom-front view of the particle were displayed every output frame, with a time interval of 0.1 ms between each frame. The video also included a dynamic plot of the instantaneous particle velocity. Another created algorithm produced a collision map of model particles. For a given particle geometry, the particle mesh was displayed in its initial position with transparent elements. The MAD simulation recorded the total duration of collision contact for each particle node, and this information was used to

create a color scale using the Jet color map in MATLAB. Particle nodes that had formed contacts were colored based on contact time. The minimum color map value was set equal to the minimum nonzero node contact time, and the maximum color map value was set equal to the maximum node contact time. For the simulations where particles were placed at varying initial centroid height, the contact time plotted for a given node was the sum of contact durations from each particle.

3.3 RESULTS

3.3.1 EFFECT OF CIRCULATING TUMOR MICROEMBOLI CONFORMATION AND SHEAR RATE ON ADHESION POTENTIAL

Trajectories were compared for model CTM particles oriented parallel to flow. A representative video of a particle trajectory, shown for a triplet-triangle, is included as a supplement (Movie S1). For a given model geometry, the centroid of the particle would oscillate towards and away from the plane wall, in a trajectory that maintained a repeating pattern dependent on the mesh configuration and initial particle orientation. The lowest point of the particle was monitored, and a collision event would be marked by a blunting of the oscillation pattern. An example of this is highlighted in Fig. 3.1, a trajectory for a 4-mer tetrahedron. In the absence of a wall interaction, the particle low point naturally oscillated as a result of the particle rotating, smoothly decreasing in height as the aggregate monomers moved towards the surface and increasing in height as the monomers moved away. During a collision, the particle would continue to translate and rotate, but being in contact with the surface, the low point could not

descend any lower. As a result, during the contact the low point height would plateau at a minimum, until the particle moved away from the surface and the low point height increased and resumed a regular pattern.

Particle trajectory was unaffected by changes in shear rate. The oscillation pattern for a given particle geometry at each shear rate was the same, with only minor differences in the local minima and maxima of centroid and low point heights. However, as the shear rate increased, the wavelength of the pattern decreased. For a given particle geometry, normalizing the simulation time by the shear rate and plotting the three trajectories on the same figure would result in the low point curves overlapping, as seen in Fig. 3.1.

Hydrodynamic behavior for all simulations was stable. The simulations persisted over an extended period of time, and in the absence of any collision interactions the model particles oscillated in a repeating pattern indefinitely, with the same approximate maximum and minimum trajectory height. This suggests that the interaction range of repulsive force chosen for the simulation was sufficiently short. An example of an extended trajectory can be seen in Fig. 3.2, shown for a model doublet.

All of the horizontally oriented particles made contact with the wall, then exhibited behavior known as “pole-vaulting.” Trajectory behavior was similar for all horizontally oriented particles, but a representative trajectory, shown for a triangular triplet, is displayed in Fig 3.3. As seen in Fig. 3.3A, fluid flow induced particle translation and rotation, which periodically brought aggregate monomers in proximity

with the plane surface. Particle velocity was proportional to centroid height, and reached a minimum when the particle formed contact with the plane surface (Fig. 3.3C). The centroid height also reached a minimum during the collision, and after colliding the particle repelled from the surface and reached a centroid height higher than its initial position (Fig. 3.3B). This change in centroid height as a result of a collision is referred to as a "pole-vault" event. After forming contact with the surface, the horizontally oriented particles all adopted new repeating trajectories at centroid heights higher than their initial positions, and they did not return to reactive distance with the surface.

Due to the orientation of the particles and the flow field, rotational motion for the rod-like and 4-mer-square aggregates occurred only about the y-axis, and points of contact were limited. Collision maps showing the cumulative node contact duration for each particle geometry are shown in Fig. 3.4. These maps correlated the cumulative frequency of contact for particles at all three observed shear rates to color intensity. Upon forming contact, the rotation of the rod-like particles decreased and contacts formed along the centerline of the particles. Particles with increased aspect ratios experienced a greater decrease in rotation upon collision, as seen by the higher maximum contact for the triplet rod vs. the doublet rod (Fig. 3.). Pole vaulting altered the particle trajectories such that minimal contact occurred on the trailing edge, and for the 4-mer square aggregate the contacts only occurred on the side of the aggregate that formed the initial collision. The 4-mer square aggregate was also the only geometry configuration to have contact occur on two monomers simultaneously.

Total contact time, maximum contact area, and time integral of contact area were quantified and compared for the different particle conformations (Fig. 3.5). As expected, the contact time for a given aggregate conformation decreased with increasing shear rate, though the maximum contact surface area remained constant (Fig 3.5A-B). The independence of maximum contact surface area on shear rate is consistent with particle trajectories being independent of shear rate. The triangular triplet had the highest total contact time, followed by the rod-like and square 4-mers. The doublet had the lowest total contact time (Fig 3.5A). The maximum contact surface area was similar for the triangular and rod-like triplets, rod-like 4-mer, and 4-mer tetrahedron. The maximum contact area was lowest for the doublet, and the 4-mer square had the highest maximum contact area by far (Fig. 3.5B). The time integral of contact area (TICA), which effectively combines contributions of collision duration with instantaneous contact area, decreased with increasing shear rate proportionately to the decrease in total contact time (Fig. 3.5C). The model doublet CTM had the lowest contact time, contact area, and TICA of any aggregate conformation. Both triplet conformations and the 4-mer rod and tetrahedron conformations had similar maximum contact areas, but the total contact time for the triplet conformation was up to about twice as long as for other conformations with similar maximum contact areas. As a result, the triangular triplet conformation displayed the highest time integral of contact area. The 4-mer square conformation had a maximum contact surface area over five times as large as any other conformation, but it had the third highest total contact time, and third highest TICA.

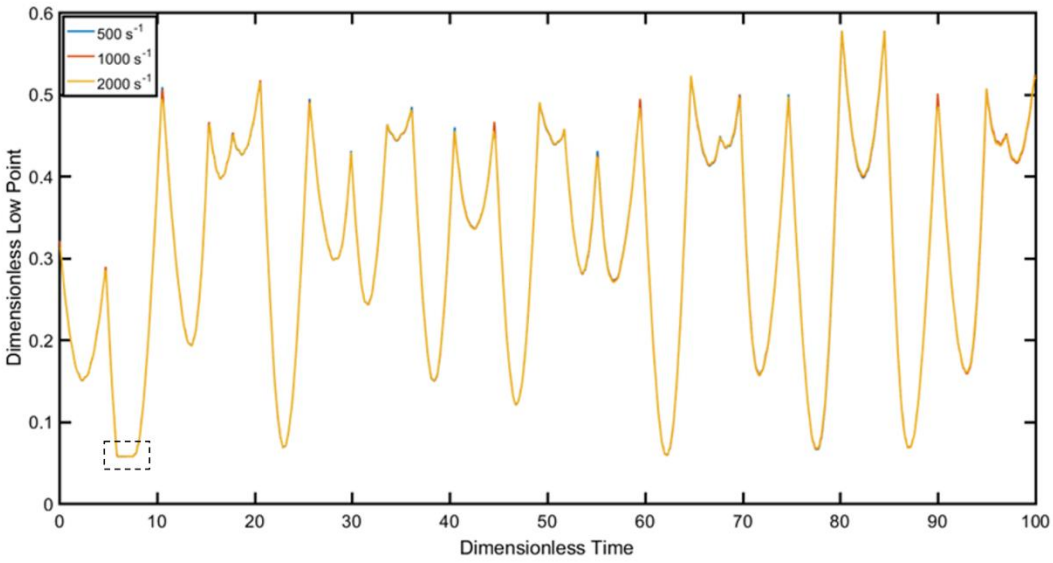

Figure 3.1: Model circulating tumor microemboli flow behavior is unaffected by shear rate. The trajectory of height of the lowest node point of tetrahedron 4-mer particles, normalized by volume-equivalent sphere radius, was plotted as a function of dimensionless time. Model CTM move in a repeating trajectory particular to its conformation and initial orientation, with the low point decreasing and increasing as aggregate monomers move towards and away from the surface. This pattern was independent of the fluid shear rate; only minor differences were observed in the local maxima and minima of low point heights. A collision event, outlined by the dashed box, corresponds with a blunting of the low point trajectory. During a collision the particle makes contact with the surface, and while the particle still translates and rotates, the low point plateaus at a minimum, until the particle moves away from the surface and resumes an oscillating pattern. For nondimensionalization, low point was normalized by volume-equivalent sphere radius and time was normalized by shear rate.

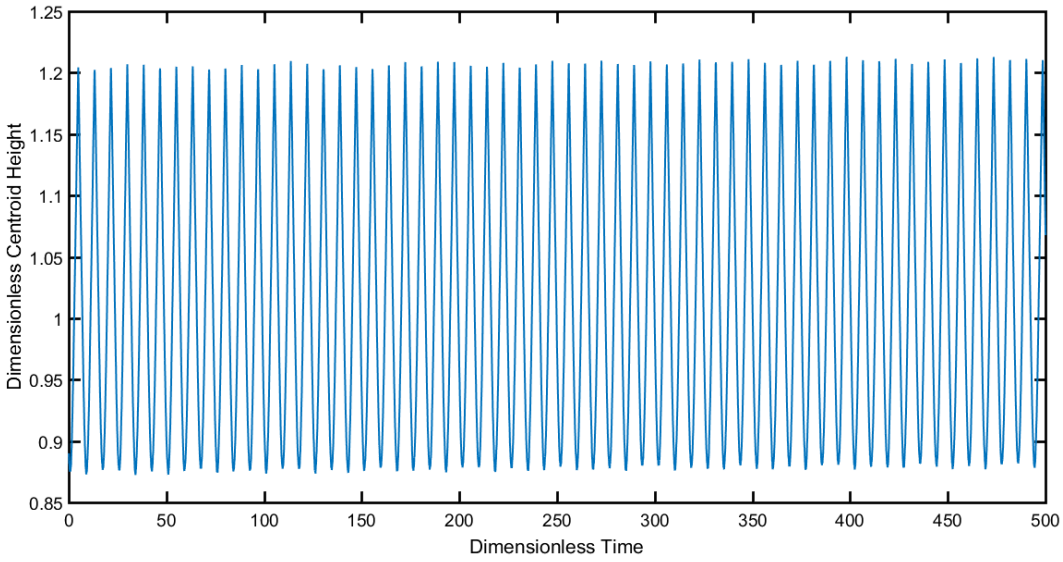

Figure 3.2: Hydrodynamic simulations are stable over extended periods of time. Dimensionless centroid height trajectory of a model CTM doublet as a function of dimensionless time. Shear rate = 1000 s^{-1}. Model particle oscillated about its initial centroid height of 1 R_e with a steady pattern for an extended period of time. For nondimensionalization, centroid height was normalized by volume-equivalent sphere radius and time was normalized by shear rate.

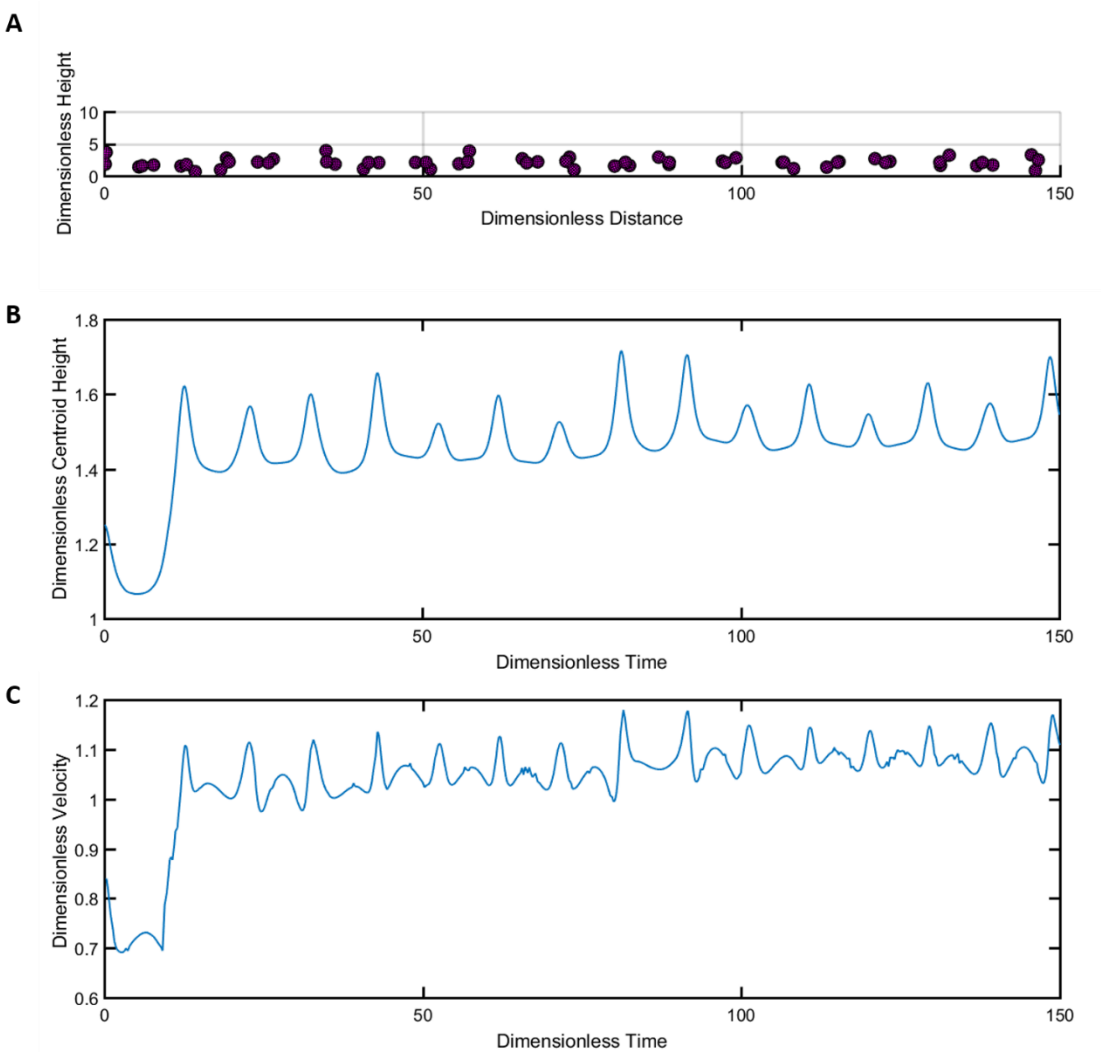

Figure 3.3: Model circulating tumor microemboli undergo collisions then exhibit pole vaulting behavior. (A) Time-lapse trajectory of a triangular triplet with an initial centroid height 0.25 R_e from the plane surface in flow, shear rate = 2000 s^{-1}. Fluid flow induced particle translation and rotation, which brought aggregate monomers within reactive distance of the plane surface. (B) Contact with the surface corresponded with the dimensionless centroid height reaching a minimum. Upon forming contact with the surface, the particle repelled, and the trajectory stabilized at a centroid height higher than its initial position. This particle motion is known as a pole vault. (C) The dimensionless velocity was proportional to centroid height, decreasing during the collision and then increasing as the particle pole vaulted. For nondimensionalization, height was normalized by volume-equivalent sphere radius, time was normalized by shear rate, and particle velocity was normalized by the fluid flow velocity at the initial centroid height.

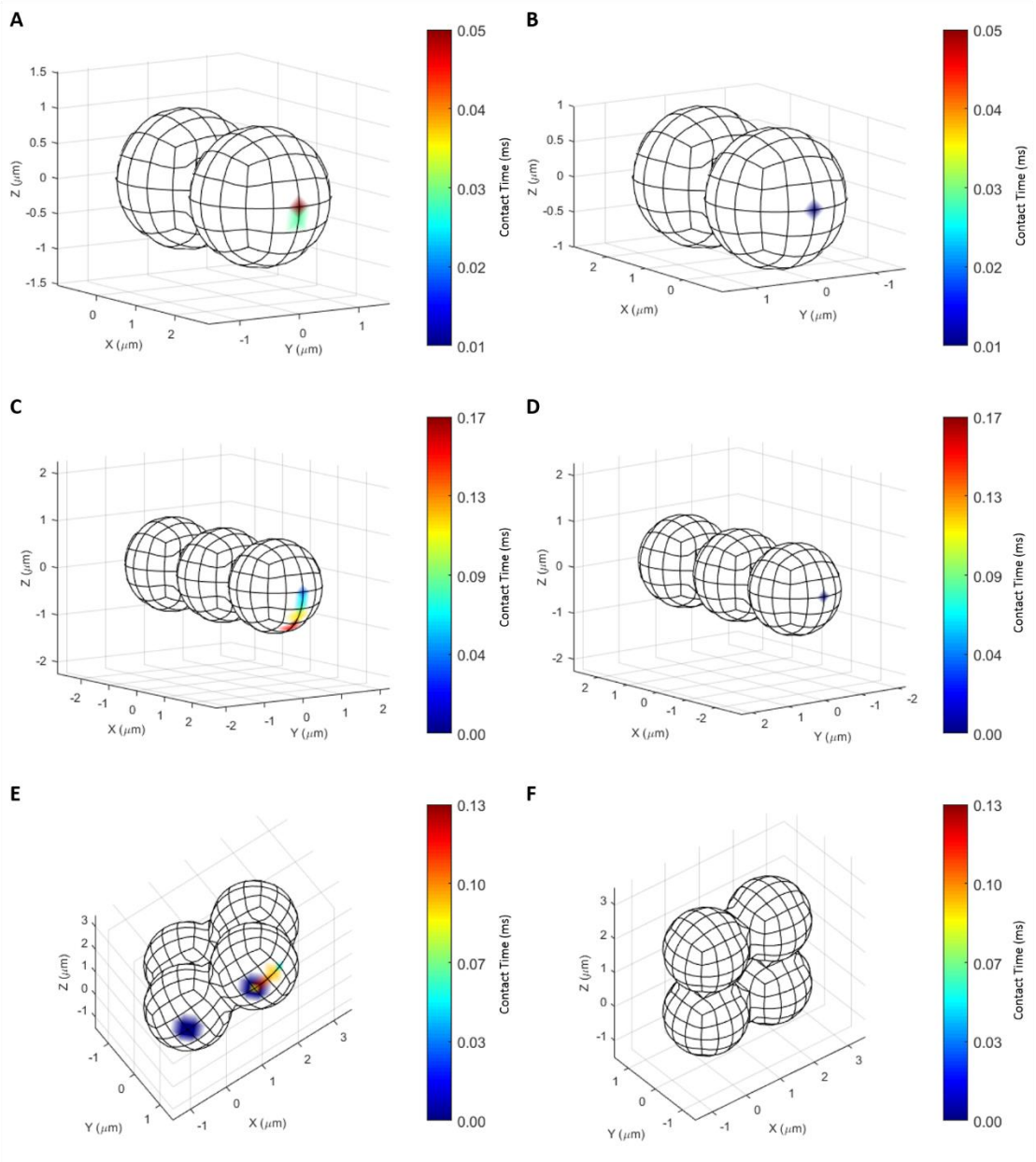

Figure 3.4: Collision maps of model circulating tumor microemboli particles. Collision maps comprising the cumulative collision events for doublet (A-B), rod-like triplet (C-D), and square 4-mer (E-F) aggregate geometries for simulations with shear rates = 500, 1000, and 2000 s^{-1}. Increased color intensity was proportional to increased total contact time. (A), (C), and (F) display contact events that occurred on the leading edge of the aggregate, and (B) and (D) display contact events that occur on the trailing edge of the aggregate. The view in (F) highlights the fact that collisions did not occur on any other edge of the particle.

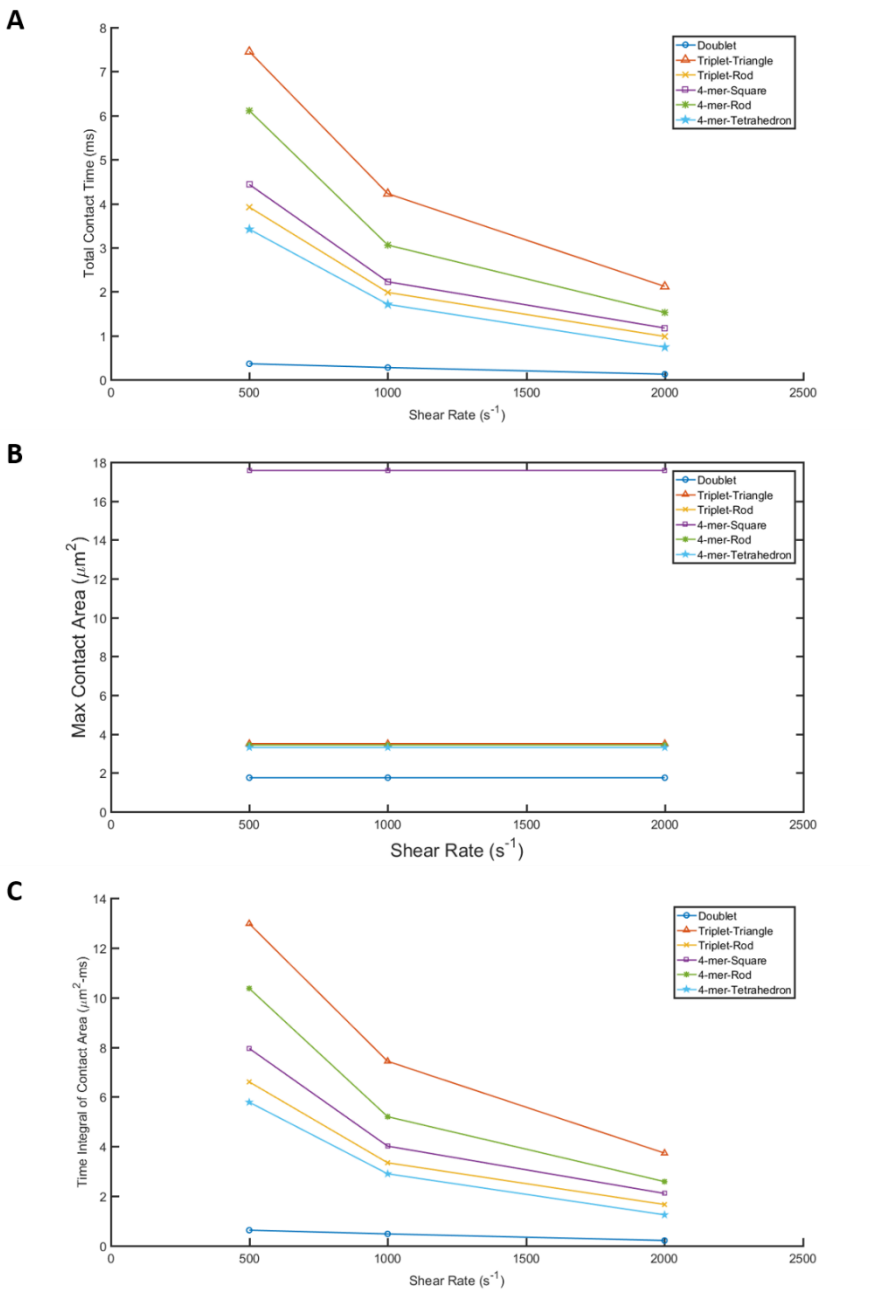

Figure 3.5: Collision efficiency is a function of shear rate and particle geometry. Comparison of total contact duration (A), maximum contact area (B), and time integral of contact area (TICA) as a function of shear rate. Increased shear rate decreases the duration of a collision, although the surface area within reactive distance of a given collision (B) remains constant. TICA (C) represents a relative measure of adhesion potential, with a higher number correlating to an increased potential to undergo a reactive event. It combines the contribution of contact area and surface area into one metric.

3.2.2 EFFECT OF CIRCULATING TUMOR MICROEMBOLI CONFORMATION AND INITIAL HEIGHT ON ADHESION POTENTIAL

Trajectories were compared for the arbitrarily oriented model CTM particles set at a range of initial heights. For each mesh geometry, I ran simulations for 3 arbitrary orientations per initial height. Fig. 3.6 displays which combinations of initial height and model particle geometry were able to form contacts with the plane wall. For combinations marked with an "X," at least one simulation in the group was capable of forming contact with the surface. Collisions only occurred for model CTM with initial centroid heights less than or equal to 0.5 R_e, and no collisions were observed for any doublet or 4-mer rod-like aggregates. When collisions did occur, pole vaulting was again observed, and particle trajectories that began at both 0.25 R_e and 0.5 R_e stabilized at around the same height. After stabilizing at the new height, model aggregates did not form collisions again. This behavior was observed for all arbitrarily oriented geometries that formed a contact; a representative illustration of this effect of initial heights on particle trajectory and collision, shown for a rod-like triplet, is displayed in Fig. 3.7.

For model geometries where collisions were observed, the total contact time, maximum contact area, and time integral of contact area (TICA) recorded for each initial orientation were averaged, and those averaged values were used to compare collision metrics between geometries. Averaged values were plotted as mean ± standard deviation. No collisions were observed for any doublet or rod-like 4-mer particles, so no collision metrics were obtained. For the particles that were able to

undergo collisions, similar contact time, maximum contact area, and time integral of contact area were observed (Fig. 3.8). The mean total contact times observed in arbitrarily oriented particles was also similar to the contact times observed for horizontally oriented particles. Total contact times for arbitrarily oriented particles ranged from 1.9 to 3.0 ms (Fig. 3.8A), and total contact times for those same particles, when initially oriented horizontally, ranged from 1.7 to 4.2 ms (Fig. 3.5A). Maximum contact areas for horizontally and arbitrarily oriented particles were similar for triplet conformations as well. For the square 4-mer, however, the contact areas differed greatly. The maximum contact area for the square 4-mer decreased from 17.574 μm^2 (Fig. 3.5B) to an average of 3.284 ± 0.319 μm^2, similar to the maximum contact areas observed for the other conformations (Fig. 3.8B). Fig. 3.8C displays the averaged TICA for the arbitrarily oriented particles, which served as a measure of relative adhesion potential. Though the rod-like triplet aggregate had the highest mean value, given the standard deviations the differences are irrelevant. Averaging the TICA values of the randomly oriented aggregates minimized the differences in adhesion potential between aggregate conformations seen when the aggregates were all horizontally oriented.

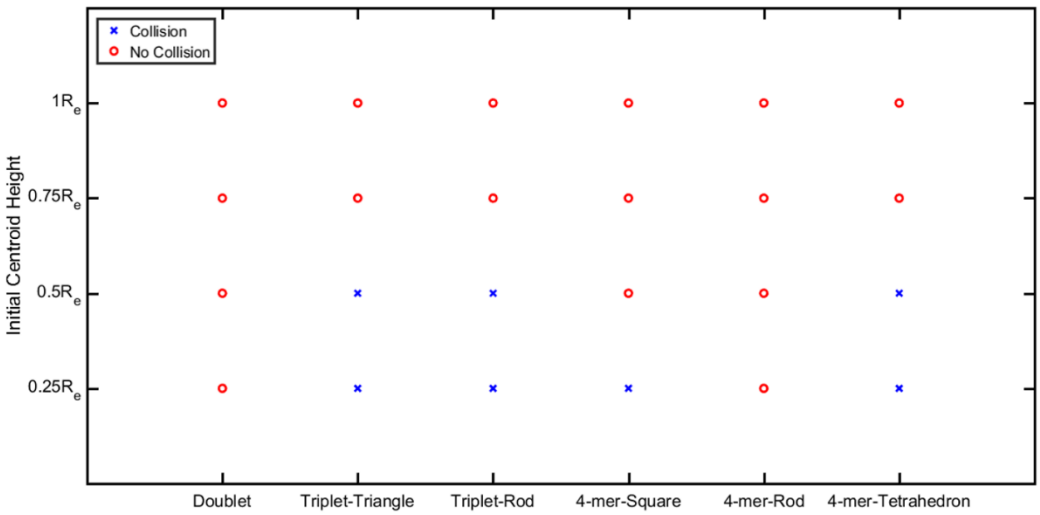

Figure 3.6: Collision occurrence for arbitrarily oriented particles. Comparison of collision occurrence for model particles, with 3 arbitrary particle orientations per initial centroid height. For initial heights marked with an "X," at least one particle in the group experienced a collision with the surface. For initial heights marked with a circle, no collisions were observed during the entirety of the simulation.

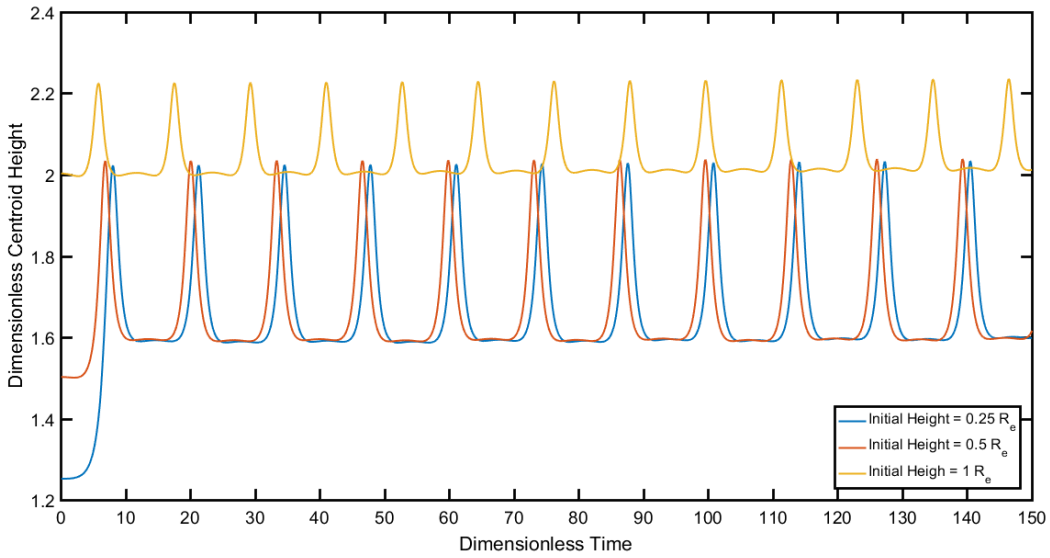

Figure 3.7: Model circulating tumor microemboli adopt stable trajectories after forming collisions. Trajectories of arbitrarily oriented rod-like triplet with initial centroid height of 0.25, 0.5, $1R_e$ freely rotating and translating in fluid with shear rate = 1000 s^{-1}. The particles with the two lowest initial heights formed collisions, which corresponded to the centroid trajectories plateauing at a minimum. Despite the trajectories reaching different minima, upon pole vaulting their trajectories stabilized at similar heights. For all particles with centroid heights above 0.5 R_e, no collisions were observed, and centroid heights oscillated with repeating patterns at consistent heights.

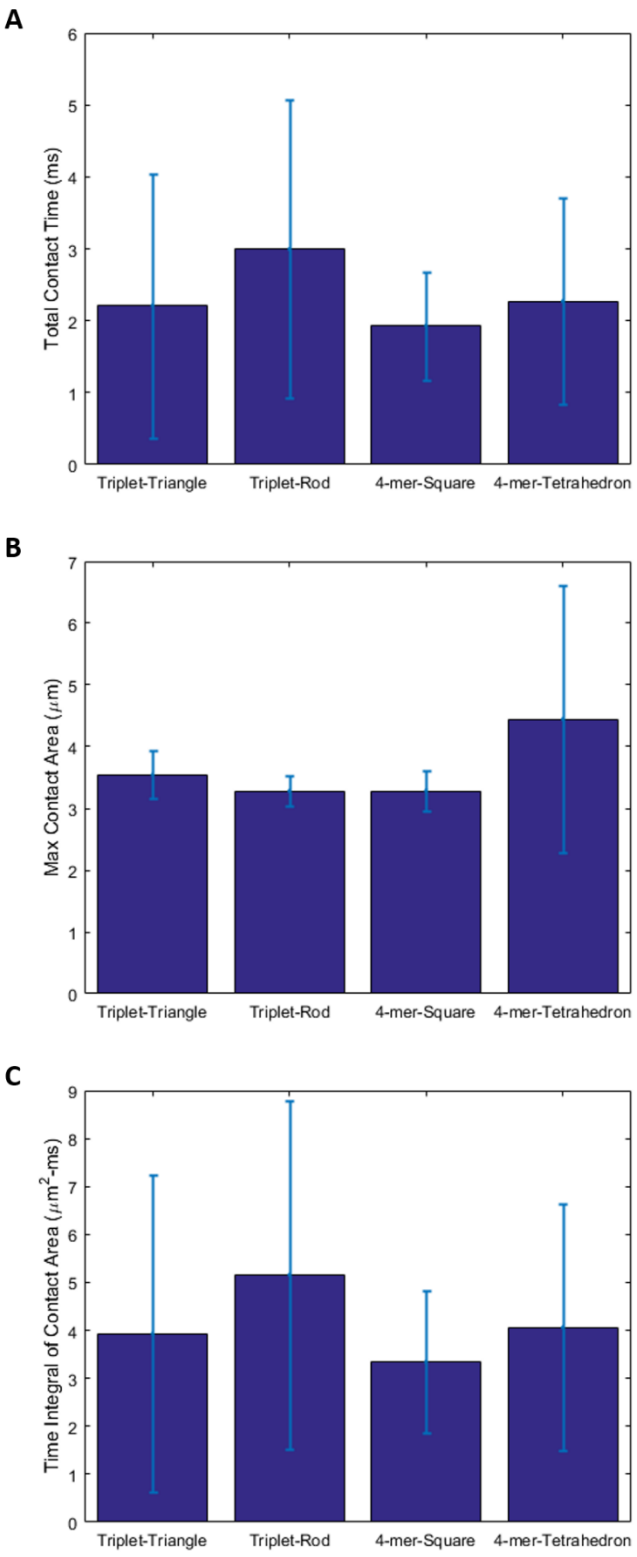

Figure 3.8: Comparison of collision metrics for arbitrarily oriented particles. For model CTM conformations where at least one arbitrarily oriented particle formed collisions with the plane surface, total contact time (A), maximum contact area (B), and mean time integral of contact area (TICA) (C) collected for each initial orientation were averaged. Average values were plotted as mean ± standard deviation. Collisions were observed only for triplet and 4-mer aggregates at initial centroid height of 0.25 or 0.5 R_e, with fluid shear rate = 1000 s^{-1} for all initial heights. The conformations that formed collisions had similar total contact times, max contact area, and adhesion probability, as indicated by similar TICA values. Averaging the collision metrics of arbitrarily oriented aggregates minimized the differences in adhesion probability seen when the triplet-triangle, triplet-rod, 4-mer-square, and 4-mer-tetrahedron were initially oriented horizontally. The doublet and rod-like 4-mer, in contrast, did not form collisions with any of the initial orientations simulated.

3.3 DISCUSSION

The MAD simulation displays excellent stability; in the absence of interactions model aggregates adopted trajectories that remained at consistent heights and continued indefinitely. Model aggregate trajectories oscillated in repeating patterns dependent on their initial height and orientation. The repeating oscillatory pattern, dependent on particle geometry, initial height, and initial orientation, is similar to the phenomenon characterized by Jeffery[118]. Pole vaulting behavior has been previously observed for numerical and experimental studies of rod-like particles, and the magnitude and frequency of this behavior is a function of nonsphericity[100, 133, 135]. As seen in Fig. 3.7, particle trajectories stabilized at a height independent of their initial centroid height. This is a result of the repulsive force, which is inversely proportional to distance. During contact, the distance reaches a minimum, and as a result the same amount of repelling force is applied to the particle. The consistency of my simulation results with prior studies suggests that appropriate values for repulsive force parameters were chosen.

As expected, when trajectory curves were plotted for a given model aggregate conformation in flow, set at the same initial height but induced by different shear rates, the curves overlapped one another once simulated time was normalized by fluid shear rate (Fig. 3.1). This can be attributed to the linearity of the Stokes equations; streamlines and fluid motion are independent of the magnitude of shear force. Also, in the absence of external forces such as gravity, wall-particle interactions are reversible. However, as seen in Fig. 3.1 the three trajectories of a 4-mer tetrahedron did not

perfectly overlap after nondimensionalization of simulated time. There were slight differences in the local maxima and minima of the oscillation pattern. This is due to the inclusion of the repulsive force interaction, which is not a function of shear rate.

When the model CTM particles were oriented parallel to the wall, every aggregate conformation tested formed contact with the plane wall. Due to the symmetrical orientation of the particles about the axes, rotation for the rod-like and square aggregates only occurred about the y-axis, and there were minimal points of contacts (Fig. 3.4). For the majority of conformations, the maximum contact area was relatively similar, but due to the shape of the square 4-mer aggregate, its maximum contact area was over 5 times that of the other conformations when the particles were initially oriented horizontally (Fig. 3.5B). Despite the square 4-mer having a much higher maximum contact area, the triangular triplet displayed the highest contact time and time integral of contact area. This may be due to the relatively intermediate sphericity of the triangular triplet. The square 4-mer and other more nonspherical geometries such as the rod-like particles, spend more time oriented parallel to flow, where any point of the particle would be sufficiently far from the wall to avoid collisions. The collision map on Fig. 3.4E-F show that contact only occurred on one side of the particle, whereas it can be seen from the triangular triplet's trajectory in Fig. 3.3A that the triplet rotated in such a way that regardless of the its orientation, a protrusion typically remained near the wall.

When the particles were given arbitrary orientations and varying initial heights, no collisions were formed for the CTM placed at 0.75 R_e or 1 R_e from the surface.

This behavior complements that seen in other studies of particles in the presence of a wall[121, 136]. Unlike the other geometries, doublet and rod-like 4-mer particles did not form collisions at any initial height. Due to constraints on computational resources, only a limited number of orientations could be run, and as a result the orientation angles represented by the experimental groups did not necessarily have a varied distribution. If enough initial orientations were simulated, it is likely that collisions would be seen for all conformations at initial centroid heights of 0.25 R_e and 0.5 R_e. As observed with the particles with fixed orientations, these geometries are capable of forming contacts with the plane surface given the proper initial orientation. As mentioned previously, the presence of a wall induces a lift force on nonspherical particles. This force is at a maximum at an orientation angle $\theta = \pi/4$, and the effect is most pronounced on particles with axis ratios ≈ 2.85[122]. This susceptibility to lift forces highlights a potentially important distinction in comparing the binding potential of cellular aggregates. The potentially increased dependence of doublet and rod-like 4-mer particles on an ideal initial orientation to form contacts implies that these conformations may have lower binding potential than the other conformations. The time integral of contact area is a useful metric for estimating relative binding potential, as it effectively combines the contributions of collision duration with instantaneous contact area. For the randomly oriented aggregates that did form geometries, the relative binding potentials, as well as the total contact time and maximum contact area, were similar. This contrasts with the horizontally oriented particles, which showed clear differences in binding potential. Interestingly, the 4-mer square particle had a

greatly decreased max contact area, but a similar binding potential, suggesting that contact time may contribute more to binding potential than the surface area. However, for these studies, more initial orientation angles would need to be evaluated to draw more definitive conclusions.

3.4 CONCLUSIONS

The hydrodynamic component of the MAD simulation successfully recapitulates the behavior of cells interacting with a plane wall, and it allows for the observation of more subtle phenomena relevant to adhesive interactions. These experiments highlighted the effect of aggregate conformation on adhesive binding potential. Conformations with a more intermediate aspect ratio and/or monomer protrusions along multiple axes, like the triangular triplet, had increased binding potential relative to doublets or 4-mer rod-like particles. The MAD simulation is particularly suited for studying these behaviors. Systematically evaluating subpopulations of CTM, like comparing collisions of triangular vs. rod-like triplet aggregates, is not feasible in physical experiments. In the observed experiments, upon forming the first collision, model aggregates pole vaulted to a height greater than their initial position, and they did not form collisions again. Though this behavior is not likely observable in *vitro* or *in vivo*, this behavior does highlight the importance of efficient binding kinetics to promote adhesion under flow. Here, contact time was driven by fluid shear rate, but in physiological environments, CTCs and CTM are brought into contact with the endothelium through collisions with other blood cells, in

a process known as margination. Total CTC and CTM contact time is driven by the rate and speed of those collisions, which are dependent on several environmental factors, like patient hematocrit level[89]. Results from simulation experiments on the effect of contact time on adhesion potential could be used to aid our understanding of the effects of certain patient characteristics on observed differences in metastatic behavior. The most computationally expensive component of the MAD simulation is the Monte Carlo simulation of receptor-ligand bonding, so the use of only the hydrodynamic component of the simulation results in faster computing times. The hydrodynamic component of the MAD simulation can thus be initially utilized as a method of evaluating a wide range of parameters to find flow conditions and particle geometries that further support or inhibit binding interactions, and the tandem use of the hydrodynamic and adhesive components of the MAD simulation would allow for more focused analysis of these conditions.

CHAPTER 4 – ADHESIVE INTERACTIONS OF CIRCULATING TUMOR MICROEMBOLI

4.1 INTRODUCTION

At present there are still challenges to developing effective isolation techniques for circulating tumor cells (CTCs), and as a result collecting samples for the study of CTCs and circulating tumor microemboli (CTM) can be difficult. CTCs are present in very low frequency in the bloodstream; counts of CTCs have estimated that there are typically less than 1 per million leukocytes. To address this issue, many isolation techniques rely on enrichment techniques, selecting samples based on a set of cell surface markers. Common isolation techniques such as CellSearch®, the only FDA-approved detection assay, target epithelial cell adhesion molecule (EpCAM), which is only present on certain subpopulations of CTCs[137-139]. Furthermore, many isolation techniques disrupt cell-cell contact and likely dissociate cell aggregates, limiting the ability of these techniques to characterize CTC aggregates[111]. In addition, these techniques may have been underestimating the prevalence of CTM. Recent advances in CTC isolation techniques have revealed evidence supporting these limitations in CTM characterization. Using traditional CTC isolation techniques such as CellSearch®, CTM are rarely found, but a technique more recently developed by Marrinucci et al. identified "HD-CTCs" without the use of protein enrichment steps, and they were able to identify CTM in 88% of prostate cancer patient samples[140]. Another microfluidic device developed by Sarioglu et al. was able to identify clusters of CTCs in 30-40% of patients with metastatic breast or prostate cancer using non-enrichment based techniques[85]. In lieu of isolation, *in vitro* studies of CTM have often had to rely on the creation of cellular aggregates through hanging drop or related

culture methods[69, 141].

The geometries used in these studies were idealized versions of those observed in experimental assays. These CTM exist transiently *in vitro* and *in vivo*. A recent study by Au et al. of CTM transit in capillaries observed that CTCs aggregate into more spherical distributions in flow, and then reorganize into single file rod-like geometries to transit through capillaries[84]. In studying the adhesive behavior of these transient aggregate conformations individually, the cumulative behavior of dynamic CTC aggregates may be better understood. Adhesive behavior can be defined under one of five general categories[78]:

1. Unbound
2. Rolling at constant speed
3. Tumbling – rolling with very brief periods of adhesion
4. Transient adhesion – rolling or tumbling with significant periods of firm adhesion
5. Firm adhesion – periods of adhesion during which the cell remains largely motionless.

Given the stochastic nature of cell adhesion, for a given set of environmental conditions a population of cells or cell clusters may exhibit multiple types of adhesive behavior. Their trajectory paths, however, seem more consistent. In our *in vitro* experiments of cell rolling, CTCs were found to translocate in very linear paths in the direction of flow. CTM, conversely, tend to translocate more perpendicularly to the flow direction. This trajectory behavior has also been seen in previous studies of CTM

migration[69].

The analysis of circulating tumor microemboli and their interactions with the endothelium is a problem particularly suited for study by the Multiparticle Adhesive Dynamics simulations. One advantage of the MAD simulation is the ability to observe adhesive interactions on the individual bond level. Many studies of binding reaction kinetics do so through the use of adhesion assays. In these experiments, the density of surface ligands is typically made low enough to assume that any interactions occurring do so through the formation of a single tether[18]. This and other assumptions must be made in order to calculate kinetic parameters. With the MAD simulation, adhesion parameters related to binding can be directly recorded. This information, coupled with data from physical experiments, may lead to more definite characterization of binding. To that end, I ran a series of simulations for model aggregates placed in close proximity to the plane surface to observe any adhesive interactions. Results were compared to *in vitro* flow assay results to validate binding kinetics parameters, and behavioral between aggregates of different sizes were compared.

4.2 METHODS

4.2.1 CELL CULTURE

The following was performed by King lab member Adelaide de Guillebon. The Colo205 cell line was obtained from American Type Culture Collection (ATCC, Manassas, VA) and cultured at 37ºC and 5% CO_2 in RPMI 1640 media (Sigma) supplemented with 10% (v/v) fetal bovine serum and 100 U/mL Penicillin-

streptomycin at 37°C in a 5% CO_2 incubator. Prior to experiments, cells were detached using StemPro Accutase (Sigma) and resuspended in media at a density of 1.5 million cells/mL. 20-24 h before cell adhesion assays, 5 µL drops of cell solution were suspended under the lid of a Petri dish filled with PBS such as to culture the cells in a "hanging drop."

4.2.2 CELL ADHESION ASSAY

The following was performed by King lab member Adelaide de Guillebon. Rectangular parallel-plate flow chambers (Glycotech) were incubated with a 10 µg/mL solution of protein G Calbiochem (Gibbstown, USA) in 1x DPBS for 1 h and 3,5, or 15 µg/mL human E-selectin/CD62E Fc chimera (R&D systems, Minneapolis, MN) for 2h, and nonspecific interactions were blocked with 5% milk in DPBS for 1 h. The flow chambers were then perfused with 1x DPBS with Ca^{2+} and Mg^{2}. Flow chambers were set onto the stage of an Olympus IX81 inverted microscope (Olympus America Inc., Melville, NY) linked to a television and Sony DVD Recorder DVO-1000MD (Sony Electronics Inc., San Diego, California). Colo205 solutions were perfused into the chamber using a syringe pump (New Era Pump Systems, Inc.) at 1 dyn/cm^2. For tracking individual rolling cells, a Proscan™II motorized stage (Prior) was used with the constant translational velocity predetermined. Rolling flux measurements and rolling velocities were then acquired using a computer-tracking program coded in MATLAB. A cell was counted as rolling if it rolled for >2 s while remaining in the field of view (432 × 324 µm2 using a 20× objective (NA, 0.40; Type,

Plan Fluorite; Olympus America Inc.) and if it translated at an average velocity less than 50% of the hydrodynamic velocity of a cell translating near the surface without forming any adhesive interactions. The hydrodynamic velocity was calculated using the theory of Goldman et al[98]. The CTM trajectory orthogonal to flow was tracked, and the angle of the centroid position relative to initial position was recorded as a function of time.

4.2.3 NUMERICAL IMPLEMENTATION

I performed simulations under the conditions listed in Table 2.2, with a shear rate of 1000 s^{-1}, for each model CTM particle at arbitrary orientations. For each size aggregate, 4-6 simulations were run, with every conformation modeled for that aggregate size included. Gravitational forces were also included, as CTM are denser than the fluid and CTM sedimentation promotes contact between the CTM and the wall. Model particles were placed with at an initial height that allowed for zero overlap between the surface roughness layers of the particle and plane wall: Initial height = particle radius + particle roughness distance + plane roughness distance + maximum bond reactive distance. Particles were allowed to translate and rotate freely for sufficient time to ensure that the results for average velocities or variance in a single model particle were not functions of simulation duration. Instantaneous position, orientation angle, time, and number of bonds existing were recorded every 100 time steps. Also recorded were the time step bonds were formed and broken and the lifespans and exerted forces for those bonds. Trajectories were analyzed using

MATLAB and ImageJ to determine the angle of CTM centroids relative to their initial position. I plotted the centroid position for the first 100 ms as a function of time using MATLAB, and I used ImageJ draw a line connecting the end centroid position with the original. The angle of that line with the centerline of flow was measured to determine the deviation angle. An example of this analysis is given in Fig. 4.1. Instantaneous translational velocity was calculated, and from that an average rolling velocity was calculated and compared to *in vitro* data. The instantaneous rolling velocity was calculated by dividing the x-distance translated during tethering by the duration of tethering. For tumbling CTM, average velocities were calculated for the periods of stable rolling. Rolling velocities and trajectories were compared to *in vitro* rolling assay data. I also developed an algorithm with MATLAB to visualize the trajectory of the CTM rolling. The algorithm produced a video of the trajectory of the particle over time, with each video frame containing a multiple views of the particle simultaneously. The side, back, and bottom-front view of the particle were displayed every output frame, with a time interval of 0.1 ms between each frame. Elements of the particle were colored to show existing bonds between the particle and plane surface. Elements were colored yellow if at least one bond had formed on that element during that output frame. Because the time scale of bond duration could be small relative to output intervals, after all bonds on an element had broken, colored elements faded back to the original particle color over the 3 subsequent output frames. The video also included dynamic plots of the instantaneous particle velocity and instantaneous bond number.

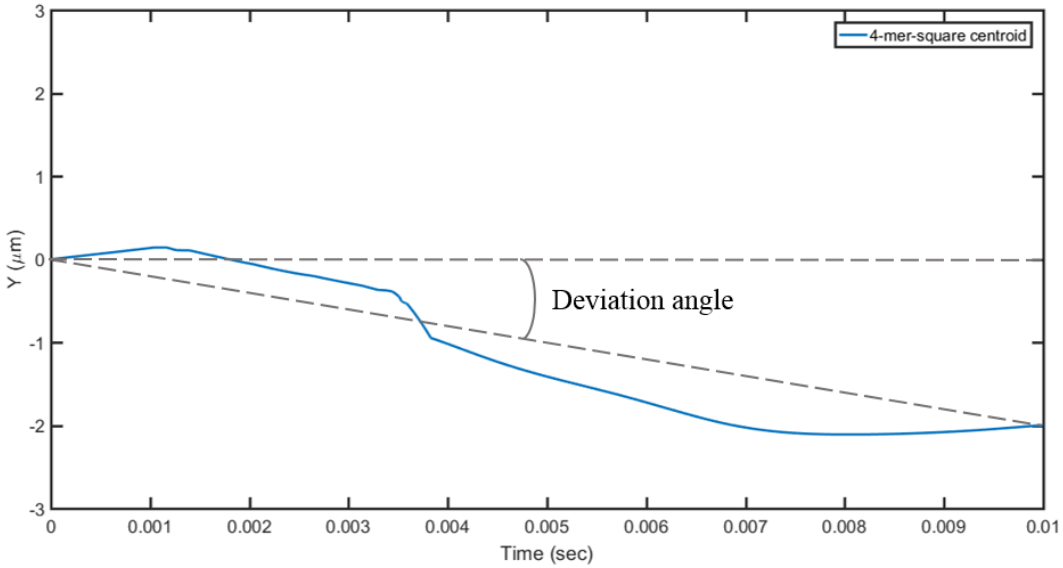

Figure 4.1: Analysis of circulating tumor microemboli deviation angle. An example of deviation angle analysis is shown for a 4-mer-square aggregate. To measure the deviation of the aggregate trajectory from the centerline, MATLAB was used to plot the centroid position over time. Using ImageJ, a line was drawn that connected the starting centroid position with the centroid position after 100 ms. A second, horizontal line was drawn extending from the initial centroid position. The angle between these two lines was measured to determine the deviation angle.

4.2.4 STATISTICAL ANALYSIS

A two-sample t-test was used for statistically analyzing the deviation angle of CTM relative to CTC. P values of less than 0.05 were considered significant.

4.3 RESULTS

4.3.1 COMPARISON OF SIMULATION RESULTS TO IN VITRO DATA

Multiple adhesive behaviors were observed during simulations of model particles. The previously defined categories of adhesive interaction behaviors did not include the more transient interactions that would be categorized as in between "unbound" and "rolling at a constant speed," or stable rolling. Those included periods of time with brief rolling interactions followed by flowing near the surface outside of bond reactive distance. Also included were particles that did not form consistent adhesive interactions, but formed brief bonds and then were repelled from the surface for variable amounts of time. These behavioral categories were included in a chart of the distribution of observed adhesive behaviors is shown in Fig. 4.2. Stable rolling was observed for one doublet, but others had inconsistent binding or very brief periods of rolling. Transient adhesion was observed for triangular triplet aggregates, and the rod-like aggregates experienced stable rolling. Square 4-mer aggregates experienced brief periods of rolling and transient adhesion. Tetrahedron 4-mer aggregates experienced brief periods of rolling and steady rolling. Rod-like 4-mer aggregates experienced inconsistent adhesive interactions.

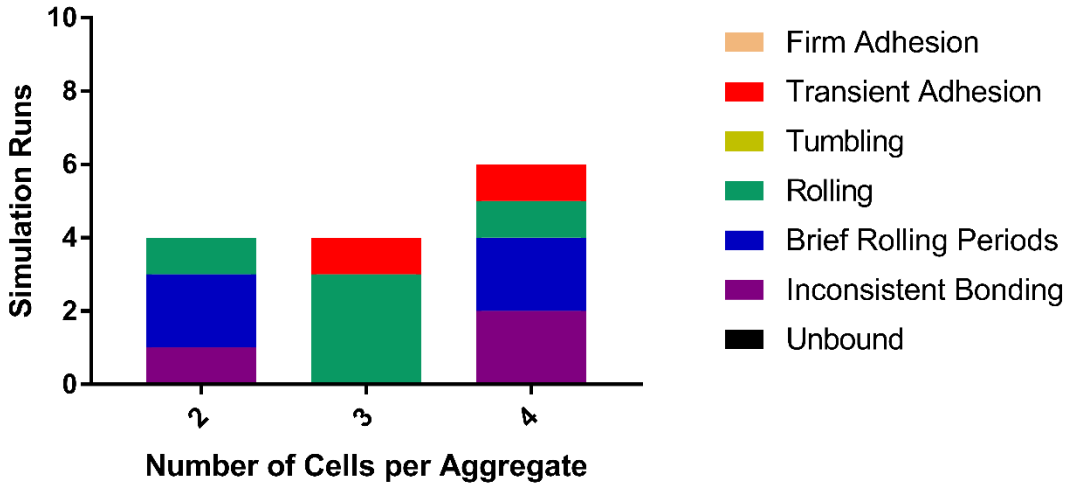

Figure 4.2: Circulating tumor microemboli in the presence of an E-selectin coated plane wall exhibit multiple adhesive behaviors. Arbitrarily oriented circulating tumor microemboli (CTM) were placed within reactive distance of an E-selectin coated plane wall. These CTM experienced multiple adhesive behaviors, and the distribution of observed behaviors was different for each size aggregate. Stable rolling was observed for one doublet, but others had inconsistent binding or very brief periods of rolling. Transient adhesion was observed for triangular triplet aggregates, and the rod-like aggregates experienced stable rolling. Square 4-mer aggregates experienced brief periods of rolling and transient adhesion. Tetrahedron 4-mer aggregates experienced brief periods of rolling and steady rolling. Rod-like 4-mer aggregates experienced inconsistent adhesive interactions.

A representative example of CTM trajectory, shown for a 4-mer square particle, is illustrated in Fig. 4.3. A representative video of particle trajectory, shown for a doublet, is included as a supplement (Movie S2). Model CTM translated and rotated in a steady trajectory, oscillating towards and away from the plane surface until reaching reactive distance and forming a bond. Initial bond formation would reduce instantaneous velocity and bring the particle closer to the plane wall, and more bonds would accumulate (Fig. 4.3A-B). If the applied load on the bonds forming on the leading edge of the aggregate was balanced by the applied load on the bonds on the trailing edge of the aggregate, the aggregate would experience stable rolling. Tumbling was marked by brief periods of bond accumulation, followed by a break in one or more bonds and an increase in low point and instantaneous velocity.

Fig. 4.4 displays the mean translocation velocities during adhesive interactions, but for the aforementioned reason rolling velocities are not plotted for the rod-like 4-mer geometry. The mean velocities of model particles had high error (standard error of 49.81, 2.12, and 20.962 μm/s for 2, 3, and 4 cell aggregates, respectively), though the results showed the qualitative trend of increased rolling velocity with aggregate size. Model aggregate particles also displayed trajectories significantly more orthogonal to flow relative to single cells, as shown in Fig. 4.5. This trend was similarly seen in the *in vitro* rolling assays. Fig 4.6 shows examples of *in vitro* trajectories. Single cells formed bonds along the leading edge of the sphere, and existing bonds were broken on the trailing edge. These bond formations and breakages caused the cell to occasionally pause or accelerate, but there was little change in lateral motion. With aggregates,

bonds could form on the leading edge of one or more monomers within the reactive distance. An uneven distribution of bonds forming or breaking on monomers in the contact area would result in the aggregate pivoting, and then rolling in a new direction. This pivot motion was previously observed for doublet yeast aggregates engineered to roll on E-selectin, and it was described as "dumbbell walking."[142].

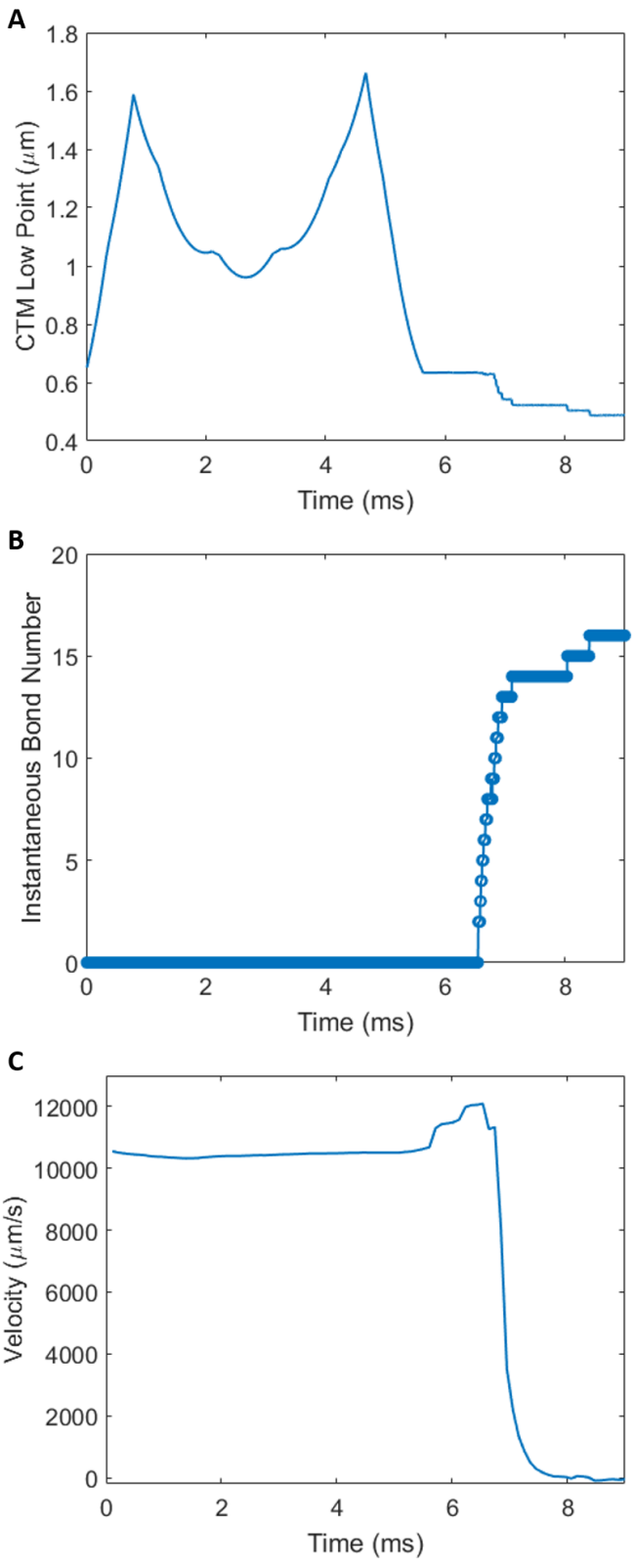

Figure 4.3: Representative circulating tumor microemboli adhesive behavior. Low point (A), instantaneous bond number (B), and velocity plot (C) for a square 4-mer model CTM particle. Typical of an adhesive interaction, CTM oscillate towards and away from surface until reaching reactive distance and forming a tether. Bonds accumulate, drawing the CTM towards the surface, and greatly decreasing its velocity.

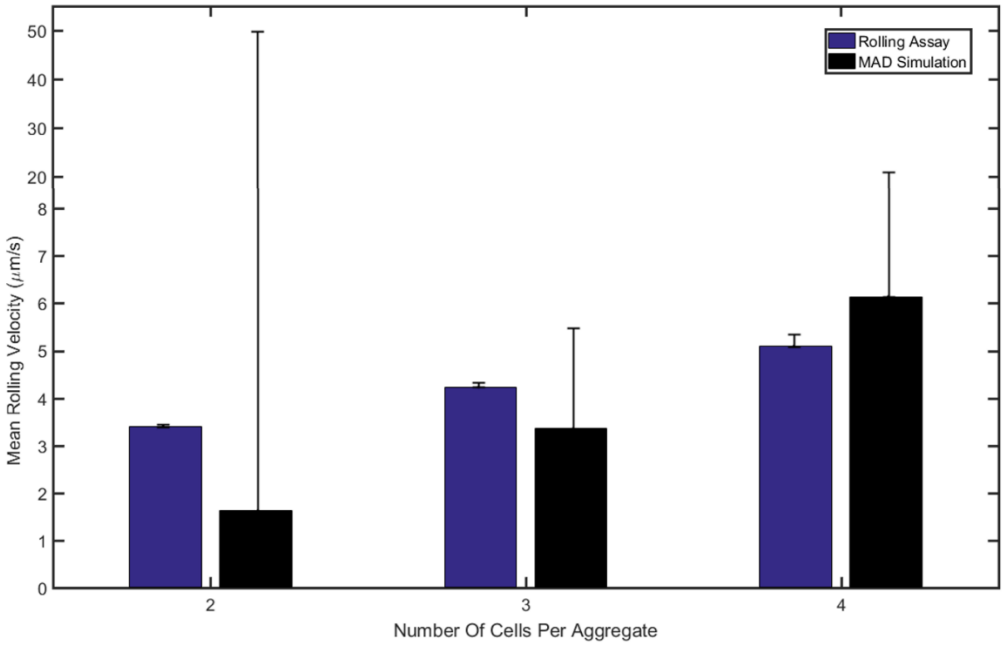

Figure 4.4: Mean rolling velocities of circulating tumor microemboli. Average of instantaneous rolling velocities collected from model and cultured Colo205 CTM, in 1000 s^{-1} shear flow. Averaged values were reported as mean ± standard error. Mean rolling velocity of Colo205 cells in a parallel plate flow chamber increased with aggregate size. Colo205 data was collected by King Lab member Adelaide de Guillebon. Results of MAD simulation show a qualitatively similar trend of cellular aggregate rolling as seen in the *in vitro* rolling assays. There is a break in the vertical axis in order to show full length of the error bars.

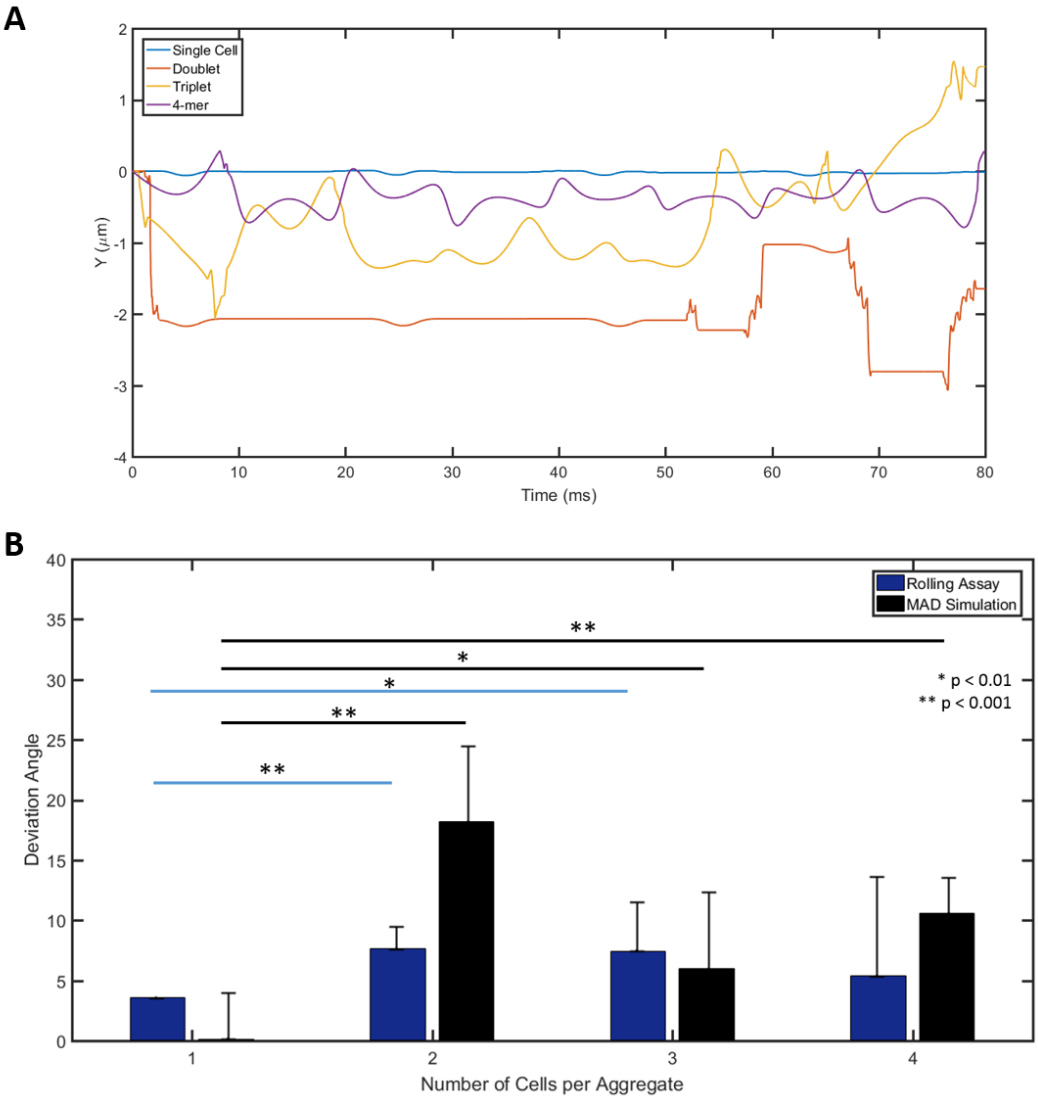

Figure 4.5: Modeled circulating tumor microemboli translocate orthogonal to direction of shear flow. A representative image of circulating tumor microemboli (CTM) translocation is shown in (A). Circulating tumor cells tend to travel in a straight path along the direction of flow. Bond formation causes pauses in motion, but there is no change in direction. Bond formation on single monomers of CTM may cause the aggregate to pivot, and upon bond breakage the CTM travels in a new direction. In (B) the trajectories of CTM were quantified. The end centroid position after 100 ms was compared to the initial centroid position of a given cell, and a straight line was drawn. The angle of that line with the centerline of flow was measured. Differences in deviation angles of cellular aggregates (>1 cells per aggregate) versus single cells were statistically compared for significance. *In vitro* data was collected by King lab member Adelaide de Guillebon.

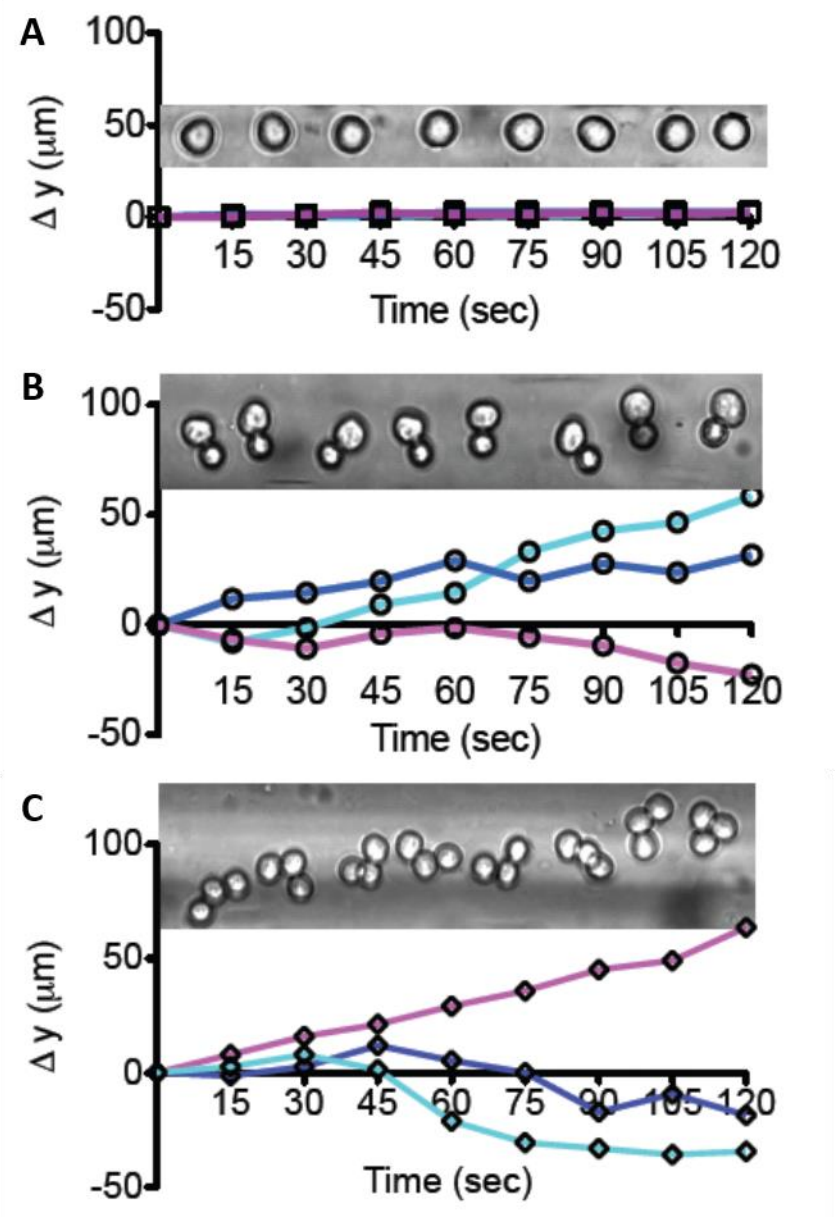

Figure 4.6: Circulating tumor cell microemboli translate orthogonal to the direction of shear flow in vitro. Representative time lapse images of a single cell, doublet aggregate, and triplet aggregate. Single cells translocated with little y-direction deviation. Doublets and triplets drifted laterally as they translated in the direction of flow. Though their lateral motions were similar relative to each other, the lateral motion of doublets and triplets was significantly higher than for single cells. Data was collected and analyzed by King lab member Adelaide de Guillebon.

The variations seen in observed adhesion behavior of CTM are reflected in the behavior of the bonds themselves. Bond lifespans increased as bonds accumulated, and the average lifespan was highest for the triangular triplet CTM that exhibited stable rolling behavior (Fig. 4.6A). The 4-mer had a smaller average bond lifespan, but it also contained the largest observed lifespans. This was due to the different adhesive interactions observed. The more spherical conformations of 4-mers experienced periods of stable rolling, mediated by fewer bonds that had much larger lifespans. The rod-like 4-mers, alternatively, tumbled with only brief adhesive interactions, and as a result had more bonds with much shorter lifespans. The rod-like 4-mer particles, however, exhibited tumbling behavior that included pole vaulting while bonds had formed (Fig 4.6B). These pole vaulting events exerted additional force on the bonds, which contributed to their increased lengths and decreased lifespans.

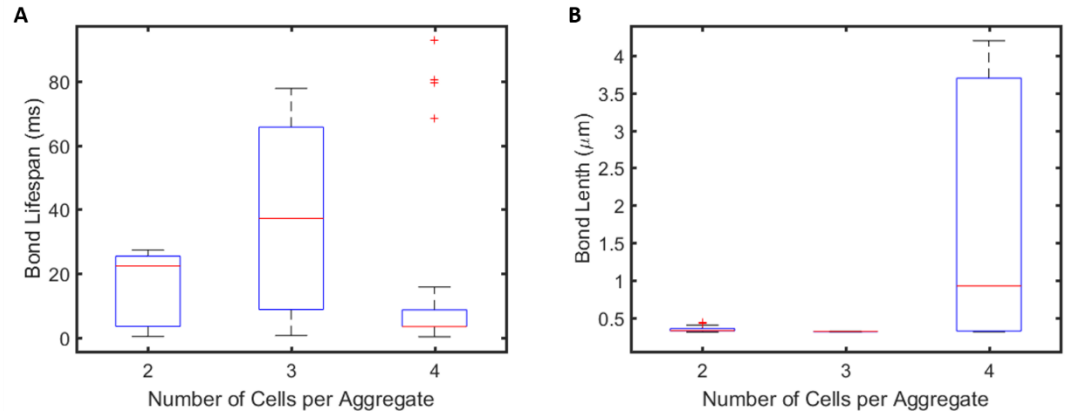

Figure 4.7: Circulating tumor microemboli adhesive interactions induce variable sLe^x/E-selectin bond behavior. Box plots of the distribution of bond lifespans and lengths for model CTM particles. Red plus signs signify outliers. The distribution of bond lengths and lifespans gives insight into the adhesive interactions that occurred during the simulations. For a given CTM, bond lifespans increased as bonds accumulated, and increased bond lengths were associated with tumbling. A narrow distribution of bond lengths with a higher average lifespan correlated with CTM experiencing phases of stable rolling. The triangular triplet confirmations exhibited stable rolling, and as a result the bond lengths remained within 7% of equilibrium bond length. The doublets experienced phases of stable rolling, but also tumbling, and as a result the distribution of bond lengths was wider than with 3-mers. 4-mer particles exhibited two distinct adhesive behaviors. The square and tetrahedron 4-mers experienced phases of stable rolling, but the rod-like 4-mer aggregates experienced tumbling without any consistent adhesive interactions. Pole vaulting events occurred for the rod-like 4-mer aggregates while bonds were formed with the surface, and these events exerted higher forces on the bonds. This resulted in bonds with much higher lengths and shorter lifespans.

4.4 DISCUSSION

The instantaneous velocity curves displayed typical rolling behavior. During the formation of tethers the CTM displayed the chaotic change in instantaneous velocity characteristic of cell rolling. A rolling cell stops when the bond sustains the load required to balance the force and torque applied on the cell by the fluid. When the bond dissociates, the velocity increases as it pivots on the newly formed bond on the leading edge of the cell, and the velocity decreases as force is accumulated in the bond. The cell decelerates if the bond can sustain the load, or it accelerates if the bond breaks prematurely[23]. This results in sharp changes in instantaneous velocity. In addition, the average velocity was determined from averaging instantaneous velocity points in the x-direction only. The time interval between velocity measurements was 1 ms; in that time span, if the applied load on the cell shifted towards the trailing edge of the cell, cell could pivot backwards temporarily, resulting in a negative instantaneous velocity measurement. Given the inherent noisiness of cell rolling, relatively high variance was expected. The *in vitro* rolling assay was used to collect data for several hundred CTCs, and the volume of samples aided in minimizing sample variance. Due to limitations in computational resources, smaller experimental groups were required, which resulted in much higher variance. The variance seen in these modeled results could be mitigated in the future by an increased sample size.

The use of sLe^x as the sole ligand for platelets and leukocytes is a simplification that must be considered when interpreting the results of the experiment. sLe^x is decorated on the end of PSGL-1, a major E-selectin binding ligand expressed

on platelets, but PSGL-1 has different binding kinetics for E-selectin than sLe^x alone[143]. Binding kinetics also differ for the L-selectin-mediated binding to E-selectin, which could potentially reflect differences in simulation results and observed leukocyte behavior *in vitro*[144]. Also the modeled cells were coated with sLe^x as their selectin ligand, as sLe^x is the terminal component of the glycoproteins that mediate colorectal cancer cell adhesion to the endothelium. However, colorectal cancer cells also highly express the isomer sLe^a. There is much more data available for the characterization of sLe^x kinetic parameters, and sLe^x and sLe^a differ only in the positioning of their fucose group, so they are often assumed to have similar kinetics[76, 114, 145]. However, given that we only measured sLe^x expression, we have likely underestimated the density of selectin ligands present on CTM, which could contribute to differences in the model results and in vitro data.

Despite these limitations, the MAD simulation was still capable of recapitulating a range of adhesive behaviors observed *in vitro*. Though there was high variance in the results, the averaged rolling velocities displayed the same qualitative trend as *in vitro* data, as rolling velocity increased with increased aggregate size. Also, model CTM exhibited trajectory paths more orthogonal to flow, behavior seen with *in vitro* CTM. This lateral movement is correlated with aggregates reaching reactive distance and forming adhesive interactions with selectins, rather than simply translocating over the selectins in close proximity but just outside of bonding range[142]. Also, this lateral drift can lead to movement over increased substrate surface area relative to single cells, increasing the potential to form adhesive interactions. Both the

in vitro and model CTM displayed angular deviations significantly different from that of CTCs. It is worth noting that the use of deviation angle is a simplified metric for comparing lateral. The CTM pivoted frequently while translocating, and particles that moved from one side of the centerline of flow to the next could have high lateral movement but display a low deviation angle, depending on their end position. However, for the experiments analyzed aggregates crossing the centerline still had end centroid sufficiently distant from the centerline to distinguish their deviation angles from those of single cells. Despites its limitations, the deviation angle still serves as a general metric to compare the lateral movement of CTM in flow.

CTM displayed stable rolling and transient adhesion, as well as more brief periods of rolling and inconsistent bonding under the flow conditions, which equate to 1 dyn/cm^2, a physiologically relevant shear stress. The rod-like 4-mer aggregate in particular resisted forming any sustained interactions with the endothelium, and many doublets only experienced brief periods of rolling. These observations are consistent with the results of experiments using the hydrodynamic component of the MAD simulation. Doublets had the lowest adhesive potential, as predicted by their time integral of contact area values, and both the doublets and rod-like 4-mer aggregates did not form collisions when randomly oriented at varying heights near the plane surface. Horizontally oriented triangular triplets had the highest adhesive potential, and randomly oriented triplet aggregates had relatively high adhesive potentials as well. Here, triplet aggregates had the most stable adhesive behaviors, and the variance in average instantaneous velocity was much lower relative to other aggregate sizes.

Due to limited computational resources, the experiments observing adhesive interactions were run only at one shear stress, 1 dyn/cm^2. This shear stress was chosen so that the experimental results could be directly compared to the results of the *in vitro* flow assays. The distribution of observed adhesive behaviors is partially dependent on fluid shear stress, so running simulations at other shear stresses would preclude drawing more firm conclusions about the differences in adhesive behaviors between aggregate sizes.

4.5 CONCLUSIONS

The MAD simulation successfully replicated adhesive behaviors observed in prior *in vitro* studies. At a physiological shear stress, rolling and adhesion interactions were observed for aggregates of all sizes observed. Aggregates rolled with significantly more lateral movement than single cells, which has implications for increased adhesion potential relative to single cells. In addition, instantaneous velocity measurements displayed the stochastic nature observed in prior experiments of selectin-mediated rolling[55, 63, 91]. Though there was a high variance in average rolling velocity, the averages increased with increasing aggregate size, a trend qualitatively similar to the *in vitro* results. The results of the averaged velocities and distributions of bond lengths and lifespans provide information about the type of adhesive interactions experienced for each aggregate size. Bond lifespans increased as bonds accumulated, and the average bond lifespan was highest for triplet aggregates. Triplet aggregates rolled with the most stability, and they had the lowest variance in average velocity.

Doublet aggregates experienced some rolling, but there were doublet aggregates incapable of forming sustained tethers with the surface. 4-mer aggregates had the most distinct differences in rolling behaviors. The rod-like 4-mer aggregates were incapable of forming sustained tethers, and pole vaulting occurred with bonds tethered, resulting in long bonds with short lifespans. The more spherical 4-mer conformations, however, exhibited rolling and transient adhesion. These results are consistent with the adhesion probabilities calculated from the experiments using the hydrodynamic component of the MAD simulation (Fig. 3.8). The inability of the rod-like 4-mer aggregates to form sustained tethers could be partially attributed to its shape and aspect ratio. A rod-like particle with a higher aspect ratio will spend more time oriented parallel to the plane surface, away from reactive distance, and it will have a higher rotation rate once oriented vertically. These behaviors likely result in limited contact time with the surface, and the increased rotation could apply a higher force on any tethers formed, limiting the potential for bond lifespan to increase. The results of these experiments suggest that, of the sizes observed, metastatic triplet aggregates have the highest potential for stable rolling and firm adhesion. They have intermediate sphericities, and the triangular triplets have protrusions that maximize the potential time the particle is within reactive distance. In order to more fully characterize CTC and CTM behavior and draw more firm conclusions, more simulations would ideally be run, encompassing a larger range of initial aggregate orientations and fluid shear rates.

CHAPTER 5 – CONCLUSIONS AND FUTURE DIRECTIONS

5.1 CONCLUSIONS OF STUDIES

I used colorectal cancer cells as a model to study the behavior of circulating tumor microemboli, particularly how their physical characteristics affect their behavior in the circulation. To that end I developed a novel adaptation of the Adhesive Dynamics simulation, referred to as the Multiparticle Adhesive Dynamics (MAD) optimized it for the interaction of Colo205 cells coated with sialyl Lewis^x (sLe^x) with an E-selectin coated plane surface. This model simulated the environment of postcapillary venules, the primary site of metastatic extravasation[146]. Aggregates of the Colo205 cell line were characterized for their size, morphology, and expression of the surface receptor sialyl Lewis^x (sLe^x). I obtained kinetic parameters for the interaction of sLe^x with E-selectin through *in vitro* experiments and the literature, and incorporated them into the simulation.

I developed model aggregates that mimicked the conformations of aggregates observed *in vitro*. The aggregates ranged in size from two to four cells, and there were 6 total conformations. The model aggregates consisted of a doublet, a triangular and rod-like triplet, and a 4-mer aggregate with a square, rod-like, and tetrahedron conformation. I used the hydrodynamic component of the MAD simulation to study the behavior of aggregates in flow and estimate their adhesive potential. The behavior of rod-like particles closely followed the behavior of ellipsoids, as observed by Jeffery[118]. When placed near the surface at a horizontal orientation, all model CTM were capable of forming collisions with the surface. Model CTM rotated, approached reactive distance with the wall, and pole vaulted, adopting a new stable trajectory

away from the wall. The duration and contact area of collision were quantified, and the time integral of contact area was used to relate the contributions of contact time and area into a measure of relative adhesion probability. Of the horizontally oriented particles, the triangular triplet had the highest adhesion probability. When oriented arbitrarily, the doublets and rod-like 4-mer aggregates were unable to form collisions for any of the initial heights or orientations tested. Arbitrarily orienting the particles minimized the differences in contact area and adhesion probability observed in the horizontally oriented aggregates. Of the arbitrarily oriented particles, the rod-like triplet had the highest average adhesion probability, though this difference was not significant.

The full MAD simulation was utilized to recapitulate the behavior of cells rolling on endothelial cells, and the observations of the simulation results were consistent with behaviors observed *in vitro*. Brief bond tethering, stable rolling, and transient adhesion were observed for the different aggregate conformations, with each size aggregate having a different distribution of observed adhesive interactions. Instantaneous rolling velocities showed the qualitatively similar stochastic nature as seen in prior studies of selectin-mediated rolling[55, 63, 68, 91]. The average velocities had high variance, but the means showed the qualitative trend of increasing velocity with increasing aggregate size seen our *in vitro* experiments. The aggregates also translocated more orthogonal to flow relative to single cells, a phenomenon also observed in our *in vitro* studies. In addition to tracking particle trajectories, the MAD simulation was able to directly record the lifespan and length of bonds that formed

during adhesive interactions. The distribution of bond lengths and bond lifespans correlated with the type of adhesive interactions that occurred. Bond lifespans increased as bonds accumulated, and a high average bond lifespan and low bond length was correlated with more stable rolling. A more narrow distribution of bond lengths correlated with similar adhesive interactions, while a larger distribution of bond lengths correlated with distinctly different types of adhesion, such as stable rolling and brief, unsubstantial tether formation. The triplet aggregates had the most stable rolling, and rod-like 4-mer particles were incapable of forming consistent tethers.

The use of this tool as a predictive model can aid in the understanding of CTM behavior in flow. The MAD simulation can be used to systematically study subpopulations of cells, and the effects of individual collisions and bond events on flow behavior and adhesive potential can be quantitatively evaluated. The results of these simulations highlight how adhesive potential can be affected by the physical characteristics of simulations alone, and those physical characteristics could potentially be linked to cellular functions.

5.2 FUTURE DIRECTONS

The MAD simulation holds great potential as a predictive tool, but there are still areas where it can be improved. The results of the rolling studies show qualitative similarities to experimental data, but given the high variance in calculated values, further optimization of kinetic parameters may still be warranted. The rolling of a

metastatic cell is sensitive to the intrinsic on-rate of bond association[78], and adjustment of that parameter alone could increase the accuracy of results. In addition, the MAD simulation employs a number of model simplifications. The model could be developed further to address these simplifications. The MAD simulation particles are rigid bodies. This approximation is reasonable for tumor cells and leukocytes prior to firm adhesion, but cell deformation has been shown to modulate selectin-mediated rolling, even prior to firm adhesion. Deformability can be incorporated into MAD simulation by modeling particles as Hookean solids with material parameters of shear modulus and Poisson's ratio. These material properties would relate force applied to a QUAD9 element to changes in its size. The calculations for deformability could be readily combined with the completed double layer boundary integral method (CDL-BIEM) to accurately model deformability for the range of shear stresses typical of postcapillary venules[147]. In addition, the use of sLe^x as a surface receptor serves as a useful start for binding mechanics relative to cancer adhesion, but ideally, integrins would need to be included to accurately model firm adhesion. The benefit of the MAD simulation is that it can model nearly any cell adhesion molecule so long as its binding kinetics are known, so incorporating another receptor would simply require an alteration to the bond probabilities, as well as the incorporation of relevant kinetic parameters. Improvements in these areas would further bolster the utility of this model in studying the range of phenomena that lead up to the formation of distant metastases.

The model aggregates used in the studies consisted of monomers permanently joined together; this was a simplification used to study the behavior of aggregates that

exist transiently *in vitro* and *in vivo*. By separately analyzing the behaviors of individual conformations that an aggregate can assume, an understanding can be developed of the overall aggregate behavior. In physiological environments, aggregates align in the most energetically favorable conformations[84]. As a result, there are environmental conditions where the rigid body conformations would not realistically exist. The rod-like 4-mer, for example, would likely exist if the major axis were oriented perpendicular to flow, and the monomers rolled downstream in unison. However it is unlikely that an aggregate would remain rod-like as multiple monomers extended away from the surface; flow forces would drive the aggregate to assume a more condensed conformation. Current MAD simulation results of those model aggregates may give added insight into why those conformations are unfavorable in physiological conditions, but for a more physiologically relevant model, an ideal aggregate model would consist of individual cells that adhere in the same stochastic manner that the cells adhere to the endothelium. Other numerical models, such as the one by developed by Rejniak, include mechanisms for homotypic aggregation, although they do not attribute the aggregation to any particular receptor-ligand pair. Galectin-3 is a protein that is overexpressed in tumor cells, and it contributes to a range of tumor functions including growth, cell adhesion, motility, and metastasis. This protein has a high affinity for the mucin-type glycoproteins that express sLe^x on colorectal cancer cells, and it has been found to mediate colorectal cancer cell homotypic adhesion[32]. Due to these factors, a galectin-mucin molecule pair could serve as the basis for homotypic aggregation of model particles. Studies of the affinity

of galectin-3 binding to mucin-type glycoproteins could be used to establish bonding probabilities for cancer cell homotypic aggregation. Using this aggregation model would allow for the identification of the preferred aggregate conformations under a given set of environmental conditions, and the adhesion efficiency of those conformations could be compared.

These experiments for the MAD simulation consisted of single aggregates interacting with the plane wall, in order to observe behaviors particular to aggregate conformations in the absence of interparticle interactions. However, these studies could be extended toward the analysis of groups of aggregates. Tumor aggregate formation can occur as a result of cells attached to the endothelium binding to cells in flow, bringing cells that would otherwise be outside of reactive distance into contact with the endothelium[30]. This process could be modeled with the inclusion of multiple particles and bond probabilities for cell-cell attachment. This process could be modeled for homotypic aggregation, and it could also extend to the aggregation of cancer cells and platelets. There is increasing evidence that the formation of platelet-tumor aggregates help evade immune responses, and the presence of these heterotypic aggregates is correlated with an increased risk of blood vessel occlusion. Platelets been previously modeled using the MAD simulation, and with the addition of currently available receptor kinetics for sLe^x/P-selectin interactions platelet-tumor aggregation could be modeled. Modeling this aggregation process could lead to a better understanding of the role of CTCs in thrombosis[12, 86].

Cancer cells are highly sensitive to their surrounding environment, and their

migration mechanisms can effectively adapt to any environmental changes. Though many studies have highlighted the significance of CTCs and their aggregates in the progression of metastasis, the physics underlying CTC transport and interactions within the vasculature remains poorly understood[69]. Developing a better understanding of CTC mechanobiology, and developing a reliable method for evaluating metastatic efficiency would be extremely valuable for developing anti-metastatic drugs. This model could be used as a preliminary method of developing more focused *in vivo* tests, which are essential to the development of anti-metastatic drugs, but are becoming increasingly expensive[40].

Currently, clinical approaches to cancer treatment often involve generalized methodology, despite the high variance between patients; this results in inappropriate or ineffective treatment. The use of tools like the MAD simulation can enable the development of more personalized treatments[49]. Most of the parameters of the MAD simulation are specific to a particular receptor-ligand pair, and others, such as surface protein expression levels, can be measured through flow cytometry or other relatively simple methods. An optimized MAD simulation could ideally be used to model patient cancer samples. One category of anti-metastatic treatment involves inhibiting the expression or functions of adhesion molecules that mediate the cascade[148-150]. However, the major difficulty with developing efficient cancer therapies is that the targets of therapies are also expressed in healthy cells. Many of these therapies rely on the fact that these target proteins are overexpressed in cancer cells relative to healthy ones, but side effects may still occur[151]. With this optimized model, it could be

possible to identify the extent of protein inhibition necessary to prevent the adhesive interactions that lead to extravasation, while minimizing patient side effects. Modeling CTC samples using an optimized MAD simulation and monitoring metastatic efficiency could serve as a noninvasive method to predict and monitor disease progression and response to therapy[49, 140].

APPENDIX

A.1 – FORTRAN 95 CODE FOR COMPLETED DOUBLE LAYER

BOUNDARY INTEGRAL METHOD

```
MODULE cdl_biem
!This module contains the subprograms required to calculate the translational and
rotational
!velocities of one or more particle of spherical or non-spherical bodies of revolution
using CDL-BIEM
!This module contains 9 subroutines + 3 subroutines (present in original MAD - point,
normal, gaussj)
!MPAD
!**********************************************************************
**************************************************

IMPLICIT NONE

integer, parameter::maxiter=500        !maximum iterations

double precision, parameter::Toler=0.0005,& !tolerance for iteration

                                mu=1.0,&!viscocity

                              quaddist=0.5,&   !distance test for quadrature order

                              rescale=1000.     !factor for avoiding round-off error


logical, parameter::shear=.true. !Determines the presence of ambient shear flow

integer::iord              !order of the Gaussian quadrature

double precision,dimension(3,3)::gpt1,gpt2,gwt       !Gaussian points and weights
(dimensioned for 3rd order)

double precision:: wt !Gaussian weight x length of normal to element

double precision,dimension(2)::eta !stores gaussian point values

double precision,dimension(3)::        x !coordinates for centroid of an element

double precision,dimension(9,3)::xq !xyz coordinates for all node points for element
iel
```

```
double precision,dimension(4)::xnorm          !unit normal of an element

!integer::difficultiter=0          !Counts the number of times in a row, the no. of
iterations is > maxiter/2


CONTAINS


!*****************************************************************
*************************************************************

SUBROUTINE GaussQuad2values(gpt1, gpt2, gwt, iord)

        double precision,dimension(3,3), INTENT(OUT)::gpt1,gpt2,gwt     !Gaussian
points and weights (dimensioned for 3rd order)

        integer, INTENT(OUT):: iord

         !Gaussian quadrature formulas

         !points and weights for curvi. quads (2nd order)

        gpt1(1,1)=-0.5773502691;      gpt2(1,1)=-0.5773502691

        gpt1(1,2)=-0.5773502691;      gpt2(1,2)=0.5773502691

        gpt1(2,1)=0.5773502691;               gpt2(2,1)=-0.5773502691

        gpt1(2,2)=0.5773502691;               gpt2(2,2)=0.5773502691

        gwt(1,1)=1.0*1.0

        gwt(1,2)=1.0*1.0

        gwt(2,1)=1.0*1.0

        gwt(2,2)=1.0*1.0

        iord=2

END SUBROUTINE GaussQuad2values
```

```
SUBROUTINE GaussQuad3values(gpt1, gpt2, gwt, iord)

        double precision,dimension(3,3), INTENT(OUT)::gpt1,gpt2,gwt     !Gaussian
points and weights (dimensioned for 3rd order)

        integer, INTENT(OUT):: iord

         !Gaussian quadrature formulas

         !points and weights for curvi. quads (3nd order)

        gpt1(1,1)=-0.7745966692;      gpt2(1,1)=-0.7745966692

        gpt1(1,2)=-0.7745966692;      gpt2(1,2)=0.0

        gpt1(1,3)=-0.7745966692;      gpt2(1,3)=0.7745966692

        gpt1(2,1)=0.0;                gpt2(2,1)=-0.7745966692

        gpt1(2,2)=0.0;                gpt2(2,2)=0.0

        gpt1(2,3)=0.0;                gpt2(2,3)=0.7745966692

        gpt1(3,1)=0.7745966692;             gpt2(3,1)=-0.7745966692

        gpt1(3,2)=0.7745966692;             gpt2(3,2)=0.0

        gpt1(3,3)=0.7745966692;             gpt2(3,3)=0.7745966692

        gwt(1,1)=0.5555555556*0.5555555556

        gwt(1,2)=0.5555555556*0.8888888889

        gwt(1,3)=gwt(1,1);      gwt(2,1)=gwt(1,2)

        gwt(2,2)=0.8888888889*0.8888888889

        gwt(2,3)=gwt(1,2);      gwt(3,1)=gwt(1,3)

        gwt(3,2)=gwt(2,3);      gwt(3,3)=gwt(1,1)

        iord=3

END SUBROUTINE GaussQuad3values
```

```
SUBROUTINE particle_mesh_properties(nsurf, nel, vcd, connect, centre, rnorml,
areas, ara, vol, ceng)

        integer, INTENT(IN):: nsurf, nel

        double precision,dimension(:,:), INTENT(IN):: vcd

        integer,dimension(:,:), INTENT(IN) ::connect

        double precision,dimension(nsurf,3), INTENT(OUT):: ceng

        double precision,dimension(nsurf), INTENT(OUT)::areas, vol !surface area
and volume of particle

        double precision,dimension(nsurf*nel,3), INTENT(OUT)::centre !element
centroid

        double precision,dimension(nsurf*nel,4), INTENT(OUT)::rnorml !unit normal
for all nel

        double precision,dimension(nsurf*nel), INTENT(OUT)::ara          !area of an
element

        integer:: iel, i, j, is, ii

        double precision::xvol,&!used for summation of cell volume

                xarea,&                 !used for summation of element area

                xdotn           !normal vector dotted with point vector

        double precision,dimension(3)::xtemp,&      !temp variable for cartesian
coordinates of element centroid

                xm              !integral measuring volume of particle?


        call GaussQuad2values(gpt1, gpt2, gwt, iord)

        do iel=1,nsurf*nel

          do i=1,9
```

```
    do j=1,3
      xq(i,j)=vcd(connect(iel,i),j)
    end do
  end do

 ! Cartesian coordinates of element centroid
  eta(1)=0.0;     eta(2)=0.0
  call point(eta, xq, xtemp)
  do i=1,3
    centre(iel,i)=xtemp(i)
  end do

  ! normal at element centroid
  call normal(eta, xq, xnorm)
  do i=1,4
    rnorml(iel,i)=xnorm(i)
  end do
end do

! calc surface area, volume and center of grav for each particle,
! also surface area of each element
do is=1,nsurf
  xvol=0.0
```

```
areas(is)=0.0
do i=1,3
  xm(i)=0.0
end do
do iel=1+(is-1)*nel,is*nel
  xarea=0.0
  do i=1,9
    do j=1,3
      xq(i,j)=vcd(connect(iel,i),j)
    end do
  end do
  do i=1,iord
    do j=1,iord
      eta(1)=gpt1(i,j);  eta(2)=gpt2(i,j)
      wt=gwt(i,j)
      call point(eta, xq, x)
      call normal(eta, xq, xnorm)
      wt=wt*xnorm(4)
      xdotn=0.0
      do ii=1,3
        xdotn=xdotn+xnorm(ii)*x(ii)
        xm(ii)=xm(ii)+x(ii)**2*xnorm(ii)*wt
```

```
            end do
            xvol=xvol+wt*xdotn
            xarea=xarea+wt
          end do
        end do
        ara(iel)=xarea
        areas(is)=areas(is)+ara(iel)
      end do
      vol(is)=abs(xvol)/3.0
      do i=1,3
        ceng(is,i)=1.5*xm(i)/xvol
      end do
    end do
END SUBROUTINE particle_mesh_properties

!**********************************************************************
**************************************************

SUBROUTINE main_cdlbiem(Gdot, nsurf, nnode, nel, step, vcd, connect, ceng, areas,
ara, centre, rnorml, sizes, extf, phi, RBMs,&
                        counter)
    integer, INTENT(IN):: nsurf, nnode, nel, step
    double precision, INTENT(IN)::Gdot
    double precision,dimension(nsurf*nnode,3), INTENT(IN):: vcd
    integer,dimension(nsurf*nel,9), INTENT(IN) ::connect
```

```
        double precision,dimension(nsurf,3), INTENT(IN):: ceng

        double precision,dimension(nsurf), INTENT(IN)::areas !surface area of
particle

        double precision, dimension(nsurf,3), INTENT(IN)::sizes    !radius of cell

        double precision,dimension(nsurf*nel,3), INTENT(IN)::centre !element
centroid

        double precision,dimension(nsurf*nel,4), INTENT(IN)::rnorml !unit normal
for all nel

        double precision,dimension(nsurf*nel), INTENT(IN)::ara    !area of an element

        double precision,dimension(nsurf,6), INTENT(INOUT)::extf !external force
OUT is specified so that variable value can be modified internally

        double precision, dimension(3*nsurf*nel),INTENT(INOUT)::phi

        double precision,dimension(nsurf,6), INTENT(OUT)::RBMs          !Rigid
body motions - translation and rotation

        integer, INTENT(INOUT):: counter

        integer:: is, j

        double precision,dimension(nsurf)::rhoxx,& !Surface integral of functions of
rho

            rhoxy,&          !Surface integral of functions of rho

            rhoxz,&          !Surface integral of functions of rho

               rhoyy,&                !Surface integral of functions of rho

            rhoyz,&          !Surface integral of functions of rho

            rhozz            !Surface integral of functions of rho

        double precision,dimension(nsurf,3)::rho      !vector from center of mass to
surface
```

```
        double precision,dimension(nsurf,6)::aa,bb,cc

        double precision, dimension(3*nsurf*nel)::bt          !RHS for Eqn 10
(PhanThien, 1992)

        double precision,dimension(nsurf*nel,3)::Snorml

        double precision,dimension(nsurf,3*nel)::phi4, phi5, phi6, Sphi4, Sphi5, Sphi6


        do is=1,nsurf

          do j=1,6

            extf(is,j)=extf(is,j)*rescale/Gdot

          end do

        end do

        call rhofunctions(nsurf, nel, vcd, connect, ceng, areas, rho, rhoxx, rhoxy,
rhoxz, rhoyy, rhoyz, rhozz)

        call null_functions(nsurf, nel, areas, centre, ceng, rho, rhoxx, rhoxy, rhoxz,
rhoyy, rhoyz, rhozz, vcd, connect, ara, &

                Snorml, phi4, phi5, phi6, Sphi4, Sphi5, Sphi6, aa, bb, cc)


        !write(45,*) 'Sphi6 =',Sphi6

        call rhs(nsurf, nel, vcd, connect, ceng, centre, rnorml, areas, extf, bt)


        !write (45,*) 'bt ',bt

        call dbl_distribution(nsurf, nel, step, rnorml, centre, sizes, vcd, connect, areas,
ara,&

                bt, Snorml, phi4, phi5, phi6, Sphi4, Sphi5, Sphi6, phi, counter)

        !write(45,*) 'phi ',phi

        call RBM_calc(Gdot, nsurf, nel, areas, ara, phi, aa, bb, cc, rho, rhoxx, rhoyy,
```

```
rhozz, &
                rhoxy, rhoyz, rhoxz, Sphi4, Sphi5, Sphi6, RBMs)


END SUBROUTINE main_cdlbiem

!**********************************************************************
**************************************************

SUBROUTINE rhofunctions(nsurf, nel, vcd, connect, ceng, areas, rho, rhoxx, rhoxy,
rhoxz, rhoyy, rhoyz, rhozz)

! geometry calculations

        integer, INTENT(IN):: nsurf, nel

        double precision,dimension(:,:), INTENT(IN):: vcd

        integer,dimension(:,:), INTENT(IN) ::connect

        double precision,dimension(nsurf,3), INTENT(IN):: ceng

        double precision,dimension(nsurf), INTENT(IN)::areas !surface area of
particle

        double precision,dimension(nsurf), INTENT(OUT)::rhoxx,&         !Surface
integral of functions of rho

            rhoxy,&         !Surface integral of functions of rho

            rhoxz,&         !Surface integral of functions of rho

              rhoyy,&               !Surface integral of functions of rho

            rhoyz,&         !Surface integral of functions of rho

            rhozz           !Surface integral of functions of rho

        double precision,dimension(nsurf,3), INTENT(OUT)::rho   !vector from center
of mass to surface

        integer ::is, iel, i, j, ii
```

```
        call GaussQuad2values(gpt1, gpt2, gwt, iord)

        rho=0.0;  rhoxx=0.0; rhoxy=0.0;  rhoxz=0.0;  rhoyy=0.0;  rhoyz=0.0;
rhozz=0.0

        do is=1,nsurf

          do iel=1+(is-1)*nel,is*nel

            do i=1,9

              do j=1,3

                xq(i,j)=vcd(connect(iel,i),j)

              end do

            end do

            do i=1,iord

              do j=1,iord

                eta(1)=gpt1(i,j);  eta(2)=gpt2(i,j)

                wt=gwt(i,j)

                call point(eta, xq, x)

                call normal(eta, xq, xnorm)

                wt=wt*xnorm(4)

                ! rho is vector from center of mass to surface

                do ii=1,3

                  x(ii)=x(ii)-ceng(is,ii)

                  rho(is,ii)=rho(is,ii)+wt*x(ii)
```

```
        end do
        rhoxx(is)=rhoxx(is)+wt*x(1)**2
        rhoxy(is)=rhoxy(is)+wt*x(1)*x(2)
        rhoxz(is)=rhoxz(is)+wt*x(1)*x(3)
        rhoyy(is)=rhoyy(is)+wt*x(2)**2
        rhoyz(is)=rhoyz(is)+wt*x(2)*x(3)
        rhozz(is)=rhozz(is)+wt*x(3)**2
      end do
    end do
  end do
  wt=1/areas(is)          !normalize with area of surface
  do i=1,3
    rho(is,i)=rho(is,i)*wt
  end do
  rhoxx(is)=rhoxx(is)*wt
  rhoxy(is)=rhoxy(is)*wt
  rhoxz(is)=rhoxz(is)*wt
  rhoyy(is)=rhoyy(is)*wt
  rhoyz(is)=rhoyz(is)*wt
  rhozz(is)=rhozz(is)*wt
end do
```

```
END SUBROUTINE rhofunctions

!**********************************************************************
**************************************************

SUBROUTINE null_functions(nsurf, nel, areas, centre, ceng, rho, rhoxx, rhoxy,
rhoxz, rhoyy, rhoyz, rhozz, vcd,&

        connect, ara, Snorml, phi4, phi5, phi6, Sphi4, Sphi5, Sphi6, aa, bb, cc)

    integer, INTENT(IN):: nsurf, nel

    double precision,dimension(:,:), INTENT(IN):: vcd

    integer,dimension(:,:), INTENT(IN) ::connect

    double precision,dimension(nsurf,3), INTENT(IN):: ceng

    double precision,dimension(nsurf), INTENT(IN)::rhoxx,&  !Surface integral of
functions of rho

        rhoxy,&         !Surface integral of functions of rho

        rhoxz,&         !Surface integral of functions of rho

          rhoyy,&               !Surface integral of functions of rho

        rhoyz,&         !Surface integral of functions of rho

        rhozz           !Surface integral of functions of rho

    double precision,dimension(nsurf,3), INTENT(IN)::rho      !vector from center
of mass to surface

    double precision,dimension(nsurf),INTENT(IN)::areas !surface area of particle

    double precision,dimension(nsurf*nel,3), INTENT(IN)::centre !element
centroid

    double precision,dimension(nsurf*nel), INTENT(IN)::ara   !area of an element

    double precision,dimension(nsurf,6), INTENT(OUT)::aa,bb,cc

    double precision,dimension(nsurf*nel,3),INTENT(OUT)::Snorml
```

```
        double precision,dimension(nsurf,3*nel),INTENT(OUT)::phi4, phi5, phi6,
Sphi4, Sphi5, Sphi6

        integer::iel,is,i,i1,i2,i3,j,ii

        double precision::tmp1, tmp20, tmp2, tmp30, tmp31, tmp3, sqrtSi,&

                phi42, phi43, phi51, phi52, phi53, phi61, phi62, phi63

        double precision,dimension(3)::xrho

        call GaussQuad2values(gpt1, gpt2, gwt, iord)

        ! coefficients of null funcs in aa, bb, cc

        aa=0.0;bb=0.0;          cc=0.0

        do is=1,nsurf

          tmp1=1.0/(sqrt(rhoyy(is)+rhozz(is)-rho(is,2)**2-rho(is,3)**2))

          aa(is,2)= rho(is,3)*tmp1

          aa(is,3)=-rho(is,2)*tmp1

          aa(is,4)=tmp1/sqrt(areas(is))

          tmp20=aa(is,3)*rho(is,1)+tmp1*rhoxy(is)

          tmp2=1.0/(sqrt(rhoxx(is)+rhozz(is)-rho(is,1)**2-rho(is,3)**2-tmp20**2))

          bb(is,1)=-tmp2*rho(is,3)

          bb(is,3)= tmp2*rho(is,1)

          bb(is,4)= tmp20*tmp2

          bb(is,5)= tmp2/sqrt(areas(is))

          tmp30= aa(is,2)*rho(is,1)-tmp1*rhoxz(is)

          tmp31=-bb(is,1)*rho(is,2)+bb(is,4)*tmp30-tmp2*rhoyz(is)
```

```
        tmp3=1.0/(sqrt(rhoxx(is)+rhoyy(is)-rho(is,1)**2-rho(is,2)**2-tmp30**2-
tmp31**2))

        cc(is,1)= tmp3*rho(is,2)

        cc(is,2)=-tmp3*rho(is,1)

        cc(is,4)=-tmp3*tmp30

        cc(is,5)=-tmp3*tmp31

        cc(is,6)= tmp3/sqrt(areas(is))

      end do


      ! calculate values of null functions at each element's centroid.

      ! first three null functions are trivial, also no container surface.

        phi4=0.0; phi5=0.0; phi6=0.0

      do is=1,nsurf

        sqrtSi=sqrt(areas(is))

        do iel=1+(is-1)*nel,is*nel

          i3=3*(iel-(is-1)*nel)

          i2=i3-1

          i1=i3-2

          do i=1,3

            xrho(i)=centre(iel,i)-ceng(is,i)

          end do

          phi42=aa(is,2)/sqrtSi-aa(is,4)*xrho(3)

          phi4(is,i2)=phi42
```

```
            phi43=aa(is,3)/sqrtSi+aa(is,4)*xrho(2)
            phi4(is,i3)=phi43
            phi51=bb(is,1)/sqrtSi+bb(is,5)*xrho(3)
            phi5(is,i1)=phi51
            phi52=bb(is,4)*phi42
            phi5(is,i2)=phi52
            phi53=bb(is,3)/sqrtSi+bb(is,4)*phi43-bb(is,5)*xrho(1)
            phi5(is,i3)=phi53
            phi6(is,i1)=cc(is,1)/sqrtSi+cc(is,5)*phi51-cc(is,6)*xrho(2)
            phi6(is,i2)=cc(is,2)/sqrtSi+cc(is,4)*phi42+cc(is,5)*phi52+cc(is,6)*xrho(1)
            phi6(is,i3)=cc(is,4)*phi43+cc(is,5)*phi53
          end do
        end do
        ! integrate null funcs and adjoint null funcs.
        Sphi4=0.0; Sphi5=0.0; Sphi6=0.0
        Snorml=0.0
isloop:do is=1,nsurf
          sqrtSi=sqrt(areas(is))
 ielloop:do iel=1+(is-1)*nel,is*nel
            i3=3*(iel-(is-1)*nel)
            i2=i3-1
            i1=i3-2
```

```
do i=1,9
  do j=1,3
    xq(i,j)=vcd(connect(iel,i),j)
  end do
end do
do i=1,iord
  do j=1,iord
    eta(1)=gpt1(i,j);  eta(2)=gpt2(i,j)
    wt=gwt(i,j)
    call point(eta, xq, x)
    call normal(eta, xq, xnorm)
    wt=wt*xnorm(4)
    do ii=1,3
      xrho(ii)=x(ii)-ceng(is,ii)
    end do
    ! null funcs evaluated at Gaussian points
    phi42=aa(is,2)/sqrtSi-aa(is,4)*xrho(3)
    phi43=aa(is,3)/sqrtSi+aa(is,4)*xrho(2)
    phi51=bb(is,1)/sqrtSi+bb(is,5)*xrho(3)
    phi52=bb(is,4)*phi42
       phi53=bb(is,3)/sqrtSi+bb(is,4)*phi43-bb(is,5)*xrho(1)
       phi61=cc(is,1)/sqrtSi+cc(is,5)*phi51-cc(is,6)*xrho(2)
```

```
      phi62=cc(is,2)/sqrtSi+cc(is,4)*phi42+cc(is,5)*phi52+cc(is,6)*xrho(1)
      phi63=cc(is,4)*phi43+cc(is,5)*phi53
      Sphi4(is,i2)=Sphi4(is,i2)+wt*phi42
      Sphi4(is,i3)=Sphi4(is,i3)+wt*phi43
      Sphi5(is,i1)=Sphi5(is,i1)+wt*phi51
      Sphi5(is,i2)=Sphi5(is,i2)+wt*phi52
      Sphi5(is,i3)=Sphi5(is,i3)+wt*phi53
      Sphi6(is,i1)=Sphi6(is,i1)+wt*phi61
      Sphi6(is,i2)=Sphi6(is,i2)+wt*phi62
      Sphi6(is,i3)=Sphi6(is,i3)+wt*phi63
      do ii=1,3
Snorml(iel,ii)=Snorml(iel,ii)+wt*xnorm(ii)
      end do
    end do
  end do
  ! normalize phi's
  Sphi4(is,i2)=Sphi4(is,i2)/ara(iel)
  Sphi4(is,i3)=Sphi4(is,i3)/ara(iel)
  do i=1,3
    Sphi5(is,i1-1+i)=Sphi5(is,i1-1+i)/ara(iel)
    Sphi6(is,i1-1+i)=Sphi6(is,i1-1+i)/ara(iel)
    Snorml(iel,i)=Snorml(iel,i)/ara(iel)
```

```
            end do

          end do ielloop

        end do isloop


END SUBROUTINE null_functions

!*****************************************************************************
*****************************************

SUBROUTINE rhs(nsurf, nel, vcd, connect, ceng, centre, rnorml, areas, extf, bt)

! calculate the right hand side of CDL-BIEM equation <----------------------

        integer, INTENT(IN):: nsurf, nel

        double precision,dimension(:,:), INTENT(IN):: vcd

        integer,dimension(:,:), INTENT(IN) ::connect

        double precision,dimension(nsurf,3), INTENT(IN):: ceng

        double precision,dimension(nsurf*nel,3), INTENT(IN)::centre !element
centroid

        double precision,dimension(nsurf),INTENT(IN)::areas !surface area of particle

        double precision,dimension(nsurf,6), INTENT(IN)::extf      !external force


        double precision, dimension(3*nsurf*nel), INTENT(OUT)::bt        !RHS for
Eqn 10 (PhanThien, 1992)

        double precision,dimension(nsurf*nel,4), INTENT(IN)::rnorml !unit normal
for all nel

        integer:: is, i, j, ii, jj, iel

        double precision:: IBdotN      !dot product of vector b with normal to element

        double precision,dimension(3)::x,&!coordinates for centroid of an element
```

```
                xp,&            !coordinates for centroid of an element
                yRHS            !temp variable for summation of RHS of eqn 10 (Phan-
Thien, 1992)
        double precision, dimension(3*nsurf*nel)::bRHS     !partial RHS for En 10
(PhanThien, 1992)


        call GaussQuad2values(gpt1, gpt2, gwt, iord)
        bt=0.0
        bRHS=0.0

        !Calculating the singularity solution in half space Eqn (13) in Phan-Thien et al,
1992
        do is=1,nsurf
          do iel=1+(is-1)*nel,is*nel
            do i=1,3
              xp(i)=centre(iel,i)
        end do
        call pointFTvelfield(xp, nsurf, ceng, extf, yRHS)
            do i=1,3
              bRHS(3*(iel-1)+i)=bRHS(3*(iel-1)+i)+yRHS(i)
            end do
          end do
        end do
```

```
! transform the RHS
do is=1,nsurf
  ! integrate b*n over surface
  IBdotN=0.0
  do iel=1+(is-1)*nel,is*nel
    do i=1,9
      do j=1,3
        xq(i,j)=vcd(connect(iel,i),j)
      end do
    end do
    do ii=1,iord
      do jj=1,iord
        eta(1)=gpt1(ii,jj);eta(2)=gpt2(ii,jj)
        wt=gwt(ii,jj)
        call point(eta, xq, x)
        call normal(eta,xq,xnorm)
        wt=wt*xnorm(4)
    call pointFTvelfield(x, nsurf, ceng, extf, yRHS)
        do i=1,3
          IBdotN=IBdotN+xnorm(i)*yRHS(i)*wt
        end do
      end do
```

```
        end do
      end do
    ! transform RHS data using psi (n/sqrt(S))
    do iel=1+(is-1)*nel,is*nel
          do i=1,3
              bt(3*(iel-1)+i)=bRHS(3*(iel-1)+i)-
0.5*IBdotN*rnorml(iel,i)/areas(is)
          end do
      end do
    end do
    ! addition to RHS due to an ambient flow
    if (shear) then
          do is=1,nsurf
            do iel=1+(is-1)*nel,is*nel
              do i=1,3
                xp(i)=centre(iel,i)
              end do
        ! shear flow, subtract ambient velocity at xp
        ! note that shear rate is rescaled to help convergence, this factor must
        ! also appear in external forces. final velocity is corrected.
        !Added later since u dot n = 0, no fluid enters the sphere
              bt(3*iel-2)=bt(3*iel-2)+rescale*xp(3)
              end do
```

```
            end do

        end if

END SUBROUTINE rhs

!*****************************************************************************
*************************************************

SUBROUTINE pointFTvelfield(xp, nsurf, ceng, extf, yRHS)

        integer, INTENT(IN):: nsurf

        double precision,dimension(nsurf,3), INTENT(IN):: ceng

        double precision, dimension(3),INTENT(IN)::xp!coordinates for centroid of an
element

        double precision,dimension(nsurf,6), INTENT(IN)::extf      !external force


        double precision, dimension(3),INTENT(OUT)::yRHS!temp variable for
summation of RHS of eqn 10 (Phan-Thien, 1992)

        integer:: i, j, k, js

        double precision:: r2,&          !square of r

            r,&                 !distance b/w particle center and element centroid

            a1,&                !coefficient multiplied with terms of single layer kernel

            rr2,&               !square of rr

               rr,&             !distance b/w particle center image and element centroid

               a2,&             !coefficient multiplied with terms of single layer kernel

               rr3,&            !cube of rr3

            t1,&                !sum of several terms of the single layer kernel

        t2,&              !sum of several terms of the single layer kernel
```

```
      PI=3.14159265358979
    double precision,dimension(3)::xc,&          !coordinates for center of a particle
       ri,&          !vector from xc to xp
      xxs,&          !image of xc
      rri,&          !vector from xxs to xp
      yRHS_T  !temp variable for summation of (T x Grad) dot G
    double precision,dimension(3,3)::d1,&  !Kronecker delta
       uk,&          !single layer kernel for unbounded flow
       uki,&         !single layer kernel for halfspace
       uke,&         !extra terms for completing the singularity soln in half-space
       Gd1,&         !gradient of G wrt x
       Gd2,&         !gradient of G wrt y
       Gd3           !gradient of G wrt z

    yRHS=0.0
 jsloop:do js=1,nsurf
        do i=1,3
         xc(i)=ceng(js,i)
         do j=1,3
          d1(i,j)=0.0
         end do
```

```
  d1(i,i)=1.0
 end do
 ! single layer kernel for unbounded solid/fluid
!calculation of distance between center of particle and element centroid x-X

 r2=0.0
 do i=1,3
  r2=r2+(xp(i)-xc(i))**2
 end do
 r=sqrt(r2)
 do i=1,3
  ri(i)=(xp(i)-xc(i))/r
 end do
 a1=1.0/(8.0*PI*mu*r)
 do i=1,3
  do j=1,3
   uk(i,j)=(ri(i)*ri(j)+d1(i,j))*a1
  end do
 end do
 ! extra terms for zero-displacement halfspace kernel
 ! calculating reflected image of point X through plane x3 = 0
 do i=1,3
```

```
  xxs(i)=xc(i)
end do
xxs(3)=-xxs(3)
!calculating distance between element centroid and image of particle center
rr2=0.0
do i=1,3
  rr2=rr2+(xp(i)-xxs(i))**2
end do
rr=sqrt(rr2)
do i=1,3
  rri(i)=(xp(i)-xxs(i))/rr
end do
! single layer kernel at the image point
a1=1.0/(8.0*PI*mu*rr)
do i=1,3
  do j=1,3
    uki(i,j)=(rri(i)*rri(j)+d1(i,j))*a1
  end do
end do
! extra terms
a1=2.0*xc(3)*a1/rr
a2=xp(3)/rr
```

```
do i=1,3
  do j=1,3
    t1=rri(i)*d1(j,3)+d1(i,3)*rri(j)-2.0*d1(i,3)*d1(j,3)*rri(3)
    t2=2.0*d1(i,3)*d1(j,3)-d1(i,j)+3.0*rri(i)*(rri(j)-2.0*d1(j,3)*rri(3))
    uke(i,j)=a1*(t1+a2*t2)
  end do
end do
do i=1,3
  do j=1,3
    uk(i,j)=uk(i,j)-uki(i,j)+uke(i,j)
    yRHS(i)=yRHS(i)-uk(i,j)*extf(js,j)
  end do
end do

! divergence of single layer kernel (to be crossed with external torque)
rr3=rr**3
a1=1.0/8.0/PI/mu
do i=1,3
  do j=1,3
    k=1
    Gd1(i,j)=-(ri(k)/r2-rri(k)/rr2)*d1(i,j)+(d1(i,k)*ri(j)+&
          d1(j,k)*ri(i))/r2-3.0*ri(i)*ri(j)*ri(k)/r2-&
```

```
      (d1(i,k)*rri(j)+d1(j,k)*rri(i))/rr2+&
      3.0*rri(i)*rri(j)*rri(k)/rr2-&
      6.0*xc(3)*rri(k)/rr3*(d1(j,3)*rri(i)+d1(i,3)*rri(j)-&
      2.0*d1(i,3)*d1(j,3)*rri(3)+xp(3)/rr*(2.0*d1(i,3)*d1(j,3)-&
      d1(i,j)+3.0*rri(i)*(rri(j)-2.0*d1(j,3)*rri(3))))+&
      2.0*xc(3)/rr3*(d1(j,3)*d1(i,k)+d1(i,3)*d1(j,k)-&
      2.0*d1(i,3)*d1(j,3)*d1(k,3)+d1(k,3)*(2.0*d1(i,3)*d1(j,3)-&
      d1(i,j)+3.0*rri(i)*(rri(j)-2.0*d1(j,3)*rri(3)))+&
      xp(3)*(3.0*d1(i,k)/rr*(rri(j)-2.0*d1(j,3)*rri(3))-&
      6.0*rri(i)*rri(k)/rr*(rri(j)-2.0*d1(j,3)*rri(3))+&
      3.0*rri(i)/rr*(d1(j,k)-2.0*d1(j,3)*d1(k,3))))
k=2
Gd2(i,j)=-(ri(k)/r2-rri(k)/rr2)*d1(i,j)+(d1(i,k)*ri(j)+&
      d1(j,k)*ri(i))/r2-3.0*ri(i)*ri(j)*ri(k)/r2-&
      (d1(i,k)*rri(j)+d1(j,k)*rri(i))/rr2+&
      3.0*rri(i)*rri(j)*rri(k)/rr2-&
      6.0*xc(3)*rri(k)/rr3*(d1(j,3)*rri(i)+d1(i,3)*rri(j)-&
      2.0*d1(i,3)*d1(j,3)*rri(3)+xp(3)/rr*(2.0*d1(i,3)*d1(j,3)-&
      d1(i,j)+3.0*rri(i)*(rri(j)-2.0*d1(j,3)*rri(3))))+&
      2.0*xc(3)/rr3*(d1(j,3)*d1(i,k)+d1(i,3)*d1(j,k)-&
      2.0*d1(i,3)*d1(j,3)*d1(k,3)+d1(k,3)*(2.0*d1(i,3)*d1(j,3)-&
      d1(i,j)+3.0*rri(i)*(rri(j)-2.0*d1(j,3)*rri(3)))+&
```

```
                xp(3)*(3.0*d1(i,k)/rr*(rri(j)-2.0*d1(j,3)*rri(3))-&
                6.0*rri(i)*rri(k)/rr*(rri(j)-2.0*d1(j,3)*rri(3))+&
                3.0*rri(i)/rr*(d1(j,k)-2.0*d1(j,3)*d1(k,3))))
          k=3
          Gd3(i,j)=-(ri(k)/r2-rri(k)/rr2)*d1(i,j)+(d1(i,k)*ri(j)+&
                d1(j,k)*ri(i))/r2-3.0*ri(i)*ri(j)*ri(k)/r2-&
                (d1(i,k)*rri(j)+d1(j,k)*rri(i))/rr2+&
                3.0*rri(i)*rri(j)*rri(k)/rr2-&
                6.0*xc(3)*rri(k)/rr3*(d1(j,3)*rri(i)+d1(i,3)*rri(j)-&
                2.0*d1(i,3)*d1(j,3)*rri(3)+xp(3)/rr*(2.0*d1(i,3)*d1(j,3)-&
                d1(i,j)+3.0*rri(i)*(rri(j)-2.0*d1(j,3)*rri(3))))+&
                2.0*xc(3)/rr3*(d1(j,3)*d1(i,k)+d1(i,3)*d1(j,k)-&
                2.0*d1(i,3)*d1(j,3)*d1(k,3)+d1(k,3)*(2.0*d1(i,3)*d1(j,3)-&
                d1(i,j)+3.0*rri(i)*(rri(j)-2.0*d1(j,3)*rri(3)))+&
                xp(3)*(3.0*d1(i,k)/rr*(rri(j)-2.0*d1(j,3)*rri(3))-&
                6.0*rri(i)*rri(k)/rr*(rri(j)-2.0*d1(j,3)*rri(3))+&
                3.0*rri(i)/rr*(d1(j,k)-2.0*d1(j,3)*d1(k,3))))
            Gd1(i,j)=Gd1(i,j)*a1
          Gd2(i,j)=Gd2(i,j)*a1
          Gd3(i,j)=Gd3(i,j)*a1
end do
    end do
```

```
            do k=1,3
              yRHS_T(k)=extf(js,5)*Gd3(k,1)-extf(js,6)*Gd2(k,1)-&
                    extf(js,4)*Gd3(k,2)+extf(js,6)*Gd1(k,2)+&
                    extf(js,4)*Gd2(k,3)-extf(js,5)*Gd1(k,3)
            end do
            do i=1,3
              yRHS(i)=yRHS(i)-yRHS_T(i)/2.0
            end do
      end do jsloop


END SUBROUTINE pointFTvelfield

!*****************************************************************************
*********************************************

SUBROUTINE dbl_distribution(nsurf, nel, step, rnorml, centre, sizes, vcd, connect,
areas, ara,&
                  bt, Snorml, phi4, phi5, phi6, Sphi4, Sphi5, Sphi6, phi, counter)
        integer, INTENT(IN):: nsurf, nel, step
        double precision,dimension(:,:), INTENT(IN):: vcd
        integer,dimension(:,:), INTENT(IN) ::connect
        double precision,dimension(nsurf*nel,3), INTENT(IN)::centre !element
centroid
        double precision,dimension(nsurf),INTENT(IN)::areas !surface area of particle
        double precision, dimension(3*nsurf*nel), INTENT(IN)::bt!RHS for Eqn 10
(PhanThien, 1992)
```

```
        double precision, dimension(nsurf,3), INTENT(IN)::sizes    !radius of cell

        double precision,dimension(nsurf*nel), INTENT(IN)::ara    !area of an element


        double precision,dimension(nsurf*nel,3), INTENT(IN)::Snorml

        double precision,dimension(nsurf*nel,4), INTENT(IN)::rnorml !unit normal
for all nel

        double precision,dimension(nsurf,3*nel), INTENT(IN)::phi4, phi5, phi6,
Sphi4, Sphi5, Sphi6

        double precision, dimension(3*nsurf*nel), INTENT(INOUT)::phi

        integer, INTENT(INOUT)::counter

        integer:: i, j, ii1, ii2, ii3, j1, iel, jj1, jj2, jj3, is, js, jstart, ii, jj, jel

        double precision::vmax,&!max bt value for normalizing the error

            rmaxerr,&

            d2,&

            eldist

        double precision,dimension(3)::x,&!coordinates for centroid of an element


              xp,&              !coordinates for centroid of an element

            srnj

        double precision,dimension(4)::rni,&

                                        xn        !stores rnorml values

        double precision, dimension(3*nsurf*nel)::phinew

        double precision,dimension(3,3)::d1,&  !Kronecker delta

              ksum,&
```

```
            ker,&
            dlker
        double precision,dimension(3,3*nsurf*nel)::zgw

        call GaussQuad2values(gpt1, gpt2, gwt, iord)
        ! begin iteration to find double layer distribution <------------------------
        ! error is normalized with max(|bt|)
        vmax=abs(bt(1))
        do i=1,nsurf*nel
          do j=1,3
            if (abs(bt(3*(i-1)+j))>vmax) then
              vmax=abs(bt(3*(i-1)+j))
            end if
          end do
        end do

        if (step==1) then !!!!.or.(counter>(maxiter/2).and.difficultiter>2)) then
!removed
          do i=1,3*nsurf*nel
            phi(i)=bt(i)  !initial guess
          end do
        end if
        rmaxerr=1.0
```

```
counter=1
do while ((rmaxerr>Toler) .and. (counter<=maxiter))
  do is=1,nsurf
    do iel=1+(is-1)*nel,is*nel
      do i=1,4
        rni(i)=rnorml(iel,i)
      end do
      do i=1,3
        xp(i)=centre(iel,i)
        do j=1,3
          ksum(i,j)=0.0
          d1(i,j)=0.0
end do
        ksum(i,i)=-1.0
        d1(i,i)=1.0
      end do
      jstart=1+(is-1)*nel
      ii3=3*(iel-jstart+1)
      ii2=ii3-1
      ii1=ii3-2
      do js=1,nsurf
        do jel=1+(js-1)*nel,js*nel
```

```
j1=3*jel-2
do i=1,3
  srnj(i)=Snorml(jel,i)
end do
ker = 0.0
dlker = 0.0
! integral of double layer kernel over element jel,
! get on-diagonal terms by summing off-diagonal terms
if (iel/=jel) then
  eldist=sqrt((centre(iel,1)-centre(jel,1))**2+&
        (centre(iel,2)-centre(jel,2))**2+&
        (centre(iel,3)-centre(jel,3))**2)
  if (eldist>quaddist*sizes(1,1).and.&
     centre(jel,3)>quaddist*sizes(1,1)) then
     wt=4.0
     do i=1,3
       x(i)=centre(jel,i)
       xn(i)=rnorml(jel,i)
     end do
     xn(4)=rnorml(jel,4)
     call dbl_calc(x, xp, xn, d1, dlker)
  else      ! use higher order quad if jel and iel are close
```

```
            do i=1,9
              do j=1,3
                xq(i,j)=vcd(connect(jel,i),j)
          end do
           end do
           dlker=0.0
           do ii=1,iord
            do jj=1,iord
              eta(1)=gpt1(ii,jj);        eta(2)=gpt2(ii,jj)
              wt=gwt(ii,jj)
              call point(eta, xq, x)
              call normal(eta,xq,xn)
              call dbl_calc(x, xp, xn, d1, dlker)
            end do
           end do
end if
if (js==is) then
  do i=1,3
    do j=1,3
      ksum(i,j)=ksum(i,j)+dlker(i,j)
    end do
  end do
```

```
  end if
end if
if (is==js) then
  d2=1/areas(js)
  jj3=3*(jel-(js-1)*nel)
  jj2=jj3-1
  jj1=jj3-2
  ker(1,1)=phi4(js,ii1)*Sphi4(js,jj1)+phi5(js,ii1)*Sphi5(js,jj1)+&
          phi6(js,ii1)*Sphi6(js,jj1)+(1.0-rni(1)*srnj(1))*d2
  ker(1,2)=phi4(js,ii1)*Sphi4(js,jj2)+phi5(js,ii1)*Sphi5(js,jj2)+&
          phi6(js,ii1)*Sphi6(js,jj2)-rni(1)*srnj(2)*d2
  ker(1,3)=phi4(js,ii1)*Sphi4(js,jj3)+phi5(js,ii1)*Sphi5(js,jj3)+&
          phi6(js,ii1)*Sphi6(js,jj3)-rni(1)*srnj(3)*d2
  ker(2,1)=phi4(js,ii2)*Sphi4(js,jj1)+phi5(js,ii2)*Sphi5(js,jj1)+&
          phi6(js,ii2)*Sphi6(js,jj1)-rni(2)*srnj(1)*d2
  ker(2,2)=phi4(js,ii2)*Sphi4(js,jj2)+phi5(js,ii2)*Sphi5(js,jj2)+&
          phi6(js,ii2)*Sphi6(js,jj2)+(1.0-rni(2)*srnj(2))*d2
  ker(2,3)=phi4(js,ii2)*Sphi4(js,jj3)+phi5(js,ii2)*Sphi5(js,jj3)+&
          phi6(js,ii2)*Sphi6(js,jj3)-rni(2)*srnj(3)*d2
  ker(3,1)=phi4(js,ii3)*Sphi4(js,jj1)+phi5(js,ii3)*Sphi5(js,jj1)+&
          phi6(js,ii3)*Sphi6(js,jj1)-rni(3)*srnj(1)*d2
  ker(3,2)=phi4(js,ii3)*Sphi4(js,jj2)+phi5(js,ii3)*Sphi5(js,jj2)+&
```

```
            phi6(js,ii3)*Sphi6(js,jj2)-rni(3)*srnj(2)*d2
      ker(3,3)=phi4(js,ii3)*Sphi4(js,jj3)+phi5(js,ii3)*Sphi5(js,jj3)+&
            phi6(js,ii3)*Sphi6(js,jj3)+(1.0-rni(3)*srnj(3))*d2
    end if
    do i=1,3
     do j=1,3
       zgw(i,j1-1+j)=ker(i,j)*ara(jel)-dlker(i,j)
     end do
    end do
  end do
end do
do i=1,3
  do j=1,3
    zgw(i,3*(iel-1)+j)=zgw(i,3*(iel-1)+j)+ksum(i,j)
  end do
end do
! update three rows
do i=1,3
  phinew(3*(iel-1)+i)=bt(3*(iel-1)+i)
  do j=1,3*nsurf*nel
    phinew(3*(iel-1)+i)=phinew(3*(iel-1)+i)-zgw(i,j)*phi(j)
  end do
```

```
            end do
            do i=3*iel-2,3*iel
              if (abs(phinew(i)-phi(i))>rmaxerr) then
                rmaxerr=abs(phinew(i)-phi(i))
              end if
            end do
          end do
        end do
        do i=1,3*nsurf*nel
          phi(i)=phinew(i)
        end do
        rmaxerr=rmaxerr/vmax
        counter=counter+1
      end do
!     if (counter>(maxiter/2)) then
!           difficultiter = difficultiter+1
!     else
!           difficultiter = 0
!     end if
END SUBROUTINE dbl_distribution
!*************************************************************************
***********************************************
```

```
SUBROUTINE dbl_calc(x, xp, xn, d1, dlker)
        double precision, dimension(3),INTENT(IN)::x, xp
        double precision, dimension(4),INTENT(IN)::xn
        double precision,dimension(3,3), INTENT(IN)::d1 !Kronecker delta
        double precision,dimension(3,3), INTENT(INOUT):: dlker
        integer::i, j
        double precision::r2,&          !square of r
            r,&             !distance b/w particle center and element centroid
            a1,&            !coefficient multiplied with terms of single/double layer
kernel
            rr2,&           !square of rr
              rr,&          !distance b/w particle center image and element centroid
            drdn,&          !dot product of ri and xn
            drrdn,&         !dot product of rri and xn
            a3,&            !coefficient multiplied with terms of double layer kernel
            a4,&            !coefficient multiplied with terms of double layer kernel
            PI=3.14159265358979
        double precision,dimension(3)::ri,&!vector from xp to x
            rri,&           !vector from xxp to x
              tt1,&
            xxp
        double precision,dimension(3,3)::tk,&!tractions/double layer kernel in half
space-Eqn 16
```

```
      t3,&          !terms for doublw layer tractions
      t4,&          !terms for double layer tractions
      tke,&         !sum of t3 and t4 multiplied by a3 and a4
      tki           !terms for double layer tractions


r2=0.0
do i=1,3
  r2=r2+(x(i)-xp(i))**2
end do
r=sqrt(r2)
drdn=0.0
do i=1,3
  ri(i)=(x(i)-xp(i))/r
  drdn=drdn+ri(i)*xn(i)
end do
a1=-3.0*drdn/(r2*4.0*PI)
do i=1,3
  do j=1,3
    tk(i,j)=ri(i)*ri(j)*a1
  end do
  xxp(i)=xp(i)
end do
```

```
xxp(3)=-xxp(3)
rr2=0.0
do i=1,3
  rr2=rr2+(x(i)-xxp(i))**2
end do
rr=sqrt(rr2)
drrdn=0.0
do i=1,3
  rri(i)=(x(i)-xxp(i))/rr
  drrdn=drrdn+rri(i)*xn(i)
end do
a1=-3.0*drrdn/(rr2*4.0*PI)
do i=1,3
  do j=1,3
    tki(i,j)=rri(i)*rri(j)*a1
  end do
end do
a3=3.0*x(3)/rr
a4=xp(3)/(rr2*rr*2.0*PI)
do i=1,3
  tt1(i)=2.0*rri(3)*d1(i,3)-rri(i)
end do
```

```
            do i=1,3
              do j=1,3
                t3(i,j)=-3.0*drrdn*rri(i)*d1(j,3)+3.0*rri(3)*xn(i)*tt1(j)
                t4(i,j)=drrdn*d1(j,i)+rri(i)*xn(j)-2.0*(drrdn*d1(i,3)+&
                      xn(3)*rri(i))*d1(j,3)+&
                      (5.0*drrdn*rri(i)-xn(i))*tt1(j)
                tke(i,j)=a4*(t3(i,j)+a3*t4(i,j))
                tk(i,j) =tk(i,j)-tki(i,j)+tke(i,j)
              end do
            end do
            wt=wt*xn(4)*2.0    ! 1st order integ. over element iel
            do i=1,3
              do j=1,3
                dlker(i,j)=dlker(i,j)+wt*tk(j,i)
              end do
            end do
END SUBROUTINE dbl_calc

!**********************************************************************
****************************************

SUBROUTINE RBM_calc(Gdot, nsurf, nel, areas, ara, phi, aa, bb, cc, rho, rhoxx,
rhoyy, rhozz, &
        rhoxy, rhoyz, rhoxz, Sphi4, Sphi5, Sphi6, RBMs)
        integer, INTENT(IN):: nsurf, nel
```

```
        double precision, INTENT(IN)::Gdot

        double precision,dimension(nsurf), INTENT(IN)::rhoxx,&  !Surface integral of
functions of rho

            rhoxy,&          !Surface integral of functions of rho

            rhoxz,&          !Surface integral of functions of rho

              rhoyy,&              !Surface integral of functions of rho

            rhoyz,&          !Surface integral of functions of rho

            rhozz            !Surface integral of functions of rho

        double precision,dimension(nsurf,3), INTENT(IN)::rho      !vector from center
of mass to surface

        double precision,dimension(nsurf),INTENT(IN)::areas !surface area of particle

        double precision,dimension(nsurf*nel), INTENT(IN)::ara    !area of an element

        double precision,dimension(nsurf,6), INTENT(IN)::aa,bb,cc

        double precision,dimension(nsurf,3*nel),INTENT(IN)::Sphi4, Sphi5, Sphi6

        double precision, dimension(3*nsurf*nel), INTENT(IN)::phi

        double precision, dimension(nsurf,6),INTENT(OUT)::RBMs

        double precision,dimension(6)::B

        integer:: i, is, jel, i1, i2, i3, j1, j2, j3

        double precision::sqrtS

        double precision,dimension(3)::xbar

        double precision,dimension(6,6)::A


        ! extract rigid body motion from the solution <----------------------------
```

```
A = 0 !Initializing A
do is=1,nsurf
 ! 1st order quadrature is used
 sqrtS=sqrt(areas(is))
 do i=1,3
  A(i,i)=sqrtS
  xbar(i)=rho(is,i)
 end do
 A(1,5)= sqrtS*xbar(3)
 A(1,6)=-sqrtS*xbar(2)
 A(2,4)=-sqrtS*xbar(3)
 A(2,6)= sqrtS*xbar(1)
 A(3,4)= sqrtS*xbar(2)
 A(3,5)=-sqrtS*xbar(1)
 A(4,2)=-aa(is,4)*areas(is)*rho(is,3)
 A(4,3)= aa(is,4)*areas(is)*rho(is,2)
 A(4,4)= aa(is,4)*areas(is)*(rhoyy(is)+rhozz(is))
 A(4,5)=-aa(is,4)*areas(is)*rhoxy(is)
 A(4,6)=-aa(is,4)*areas(is)*rhoxz(is)
 A(5,1)= bb(is,5)*areas(is)*rho(is,3)
 A(5,3)=-bb(is,5)*areas(is)*rho(is,1)
 A(5,4)=-bb(is,5)*areas(is)*rhoxy(is)
```

```
A(5,5)= bb(is,5)*areas(is)*(rhoxx(is)+rhozz(is))
A(5,6)=-bb(is,5)*areas(is)*rhoyz(is)
A(6,1)=-cc(is,6)*areas(is)*rho(is,2)
A(6,2)= cc(is,6)*areas(is)*rho(is,1)
A(6,4)=-cc(is,6)*areas(is)*rhoxz(is)
A(6,5)=-cc(is,6)*areas(is)*rhoyz(is)
A(6,6)= cc(is,6)*areas(is)*(rhoxx(is)+rhoyy(is))

!write(45,*) 'A =', A
do i=1,6
  B(i)=0.0
end do
do jel=1+(is-1)*nel,is*nel
  do i=1,3
    B(i)=B(i)+phi(3*(jel-1)+i)*ara(jel)
  end do
end do

do i=1,3
  B(i)=B(i)/sqrtS
end do
do jel=1+(is-1)*nel,is*nel
```

```
  i3=3*jel
  i2=i3-1
  i1=i3-2
  j3=3*(jel-(is-1)*nel)
  j2=j3-1
  j1=j3-2
  B(4)=B(4)+(phi(i2)*Sphi4(is,j2)+phi(i3)*Sphi4(is,j3))*ara(jel)
  B(5)=B(5)+(phi(i1)*Sphi5(is,j1)+phi(i2)*Sphi5(is,j2)+&
           phi(i3)*Sphi5(is,j3))*ara(jel)
  B(6)=B(6)+(phi(i1)*Sphi6(is,j1)+phi(i2)*Sphi6(is,j2)+&
           phi(i3)*Sphi6(is,j3))*ara(jel)
 end do
 B(6)=B(6)-cc(is,1)*B(1)-cc(is,2)*B(2)-cc(is,4)*B(4)-cc(is,5)*B(5)
 B(5)=B(5)-bb(is,1)*B(1)-bb(is,3)*B(3)-bb(is,4)*B(4)
 B(4)=B(4)-aa(is,2)*B(2)-aa(is,3)*B(3)

 !write(45,*) 'B =',B

 ! solve A*x=B via Gauss-Jordan elimination
 call gaussj(A,6,6,B,1,1)

do i=1,6
  RBMs(is,i)=B(i)/rescale*(Gdot)
```

```
  end do
 end do
 !write(45,*) 'RBMs =',RBMs
END SUBROUTINE RBM_calc
!*****************************************************************************************************
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE calcceng(vcd, connect, nel, nsurf, tempceng)
 integer, INTENT(IN):: nsurf, nel
 double precision,dimension(:,:), INTENT(IN):: vcd
 integer,dimension(:,:), INTENT(IN) ::connect
 double precision,dimension(nsurf,3), INTENT(OUT):: tempceng
 integer::is,iel,i,j,ii
 double precision::xvol,wt,xdotn
 double precision,dimension(3)::xm,x
 double precision,dimension(4)::xnorm

 call GaussQuad2values(gpt1, gpt2, gwt, iord)
 do is=1,nsurf
  xvol=0.0
  xm=0.0
  do iel=1+(is-1)*nel,is*nel
```

```
 do i=1,9
  do j=1,3
    xq(i,j)=vcd(connect(iel,i),j)
  end do
 end do

 do i=1,iord
  do j=1,iord
    eta(1)=gpt1(i,j);  eta(2)=gpt2(i,j)
    wt=gwt(i,j)
    call point(eta, xq, x)
    call normal(eta, xq, xnorm)
    wt=wt*xnorm(4)
    xdotn=0.0
    do ii=1,3
      xdotn=xdotn+xnorm(ii)*x(ii)
      xm(ii)=xm(ii)+x(ii)**2*xnorm(ii)*wt
    end do
    xvol=xvol+wt*xdotn
  end do
 end do
end do
do i=1,3
```

```
        tempceng(is,i)=1.5*xm(i)/xvol
       end do
      end do
END SUBROUTINE calcceng
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE point(eta, xq, x)
integer::i,j
double precision,dimension(9,3), INTENT(IN)::xq
double precision,dimension(2), INTENT(IN)::eta
double precision::opr,omr,ops,oms,s1,s2,omrr,omss
double precision,dimension(3), INTENT(OUT)::x
double precision,dimension(9)::shape

opr=1.0+eta(1);     omr=1.0-eta(1);     ops=1.0+eta(2);     oms=1.0-eta(2)
do i=1,9
  shape(i)=0.0
end do
shape(1)=omr*oms/4.0
shape(2)=opr*oms/4.0
shape(3)=opr*ops/4.0
shape(4)=omr*ops/4.0
omrr=1.0-eta(1)**2;   omss=1.0-eta(2)**2
```

```
shape(5)=omrr*oms/2.0
shape(6)=omss*opr/2.0
shape(7)=omrr*ops/2.0
shape(8)=omss*omr/2.0
shape(1)=shape(1)-shape(5)/2.0-shape(8)/2.0
do i=2,4
  shape(i)=shape(i)-shape(i+3)/2.0-shape(i+4)/2.0
end do
shape(9)=omrr*omss
s1=shape(9)/4.0;         s2=shape(9)/2.0
do i=1,4
  shape(i)=shape(i)+s1
  shape(i+4)=shape(i+4)-s2
end do
do i=1,3
  x(i)=0.0
end do
do i=1,9
  do j=1,3
    x(j)=x(j)+shape(i)*xq(i,j)
  end do
end do
```

```
END SUBROUTINE point

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

SUBROUTINE normal(eta, xq, x)

      integer::i,j

      double precision::s1,s2,s3,s4,x2

      double precision,dimension(2), INTENT(IN)::eta

      double precision,dimension(9,3), INTENT(IN)::xq

      double precision,dimension(4), INTENT(OUT)::x

      double precision,dimension(9,2)::deriv

      double precision,dimension(3,3)::xjac


      do i=1,9

        deriv(i,1)=0.0

        deriv(i,2)=0.0

      end do

      deriv(1,1)=(eta(2)-1.0)/4.0

      deriv(2,1)=-deriv(1,1)

      deriv(3,1)=(eta(2)+1.0)/4.0

      deriv(4,1)=-deriv(3,1)

      deriv(1,2)=(eta(1)-1.0)/4.0

      deriv(2,2)=(-eta(1)-1.0)/4.0

      deriv(3,2)=-deriv(2,2)
```

```
deriv(4,2)=-deriv(1,2)
deriv(5,1)=(eta(2)-1.0)*eta(1)
deriv(6,1)=(1.0-eta(2)**2)/2.0
deriv(7,1)=(-eta(2)-1.0)*eta(1)
deriv(8,1)=-deriv(6,1)
deriv(1,1)=deriv(1,1)-deriv(5,1)/2.0-deriv(8,1)/2.0
do i=2,4
  deriv(i,1)=deriv(i,1)-deriv(i+3,1)/2.0-deriv(i+4,1)/2.0
end do
deriv(5,2)=(eta(1)**2-1.0)/2.0
deriv(6,2)=(-eta(1)-1.0)*eta(2)
deriv(7,2)=-deriv(5,2)
deriv(8,2)=(eta(1)-1.0)*eta(2)
deriv(1,2)=deriv(1,2)-deriv(5,2)/2.0-deriv(8,2)/2.0
do i=2,4
  deriv(i,2)=deriv(i,2)-deriv(i+3,2)/2.0-deriv(i+4,2)/2.0
end do
deriv(9,1)=2.0*eta(1)*(eta(2)**2-1.0)
deriv(9,2)=2.0*eta(2)*(eta(1)**2-1.0)
s1=deriv(9,1)/4.0;      s2=s1*2.0;      s3=deriv(9,2)/4.0;      s4=s3*2.0
do i=1,4
  deriv(i,1)=deriv(i,1)+s1
```

```
  deriv(i+4,1)=deriv(i+4,1)-s2
  deriv(i,2)=deriv(i,2)+s3
  deriv(i+4,2)=deriv(i+4,2)-s4
end do

do i=1,3
  do j=1,3
    xjac(i,j)=0.0
  end do
end do
do i=1,9
  do j=1,3
    xjac(1,j)=xjac(1,j)+deriv(i,1)*xq(i,j)
    xjac(2,j)=xjac(2,j)+deriv(i,2)*xq(i,j)
  end do
end do
xjac(3,1)=xjac(1,2)*xjac(2,3)-xjac(1,3)*xjac(2,2)
xjac(3,2)=xjac(1,3)*xjac(2,1)-xjac(1,1)*xjac(2,3)
xjac(3,3)=xjac(1,1)*xjac(2,2)-xjac(1,2)*xjac(2,1)

x2=0.0
do i=1,3
```

```
    x(i)=xjac(3,i)
    x2=x2+x(i)**2
  end do
  x(4)=sqrt(x2)
  do i=1,3
    x(i)=x(i)/x(4)
  end do

END SUBROUTINE normal
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SUBROUTINE gaussj(a,n,np,b,m,mp) ! from Numerical Recipes
integer, INTENT(IN)::m,mp,n,np
integer,parameter::nmax=50
double precision,dimension(np,np), INTENT(INOUT)::a
double precision,dimension(np,mp), INTENT(INOUT)::b
integer::i,icol,irow,j,k,l,ll
integer,dimension(nmax)::indxc,indxr,ipiv
double precision::big,dum,pivinv

do j=1,n
  ipiv(j)=0
end do
do i=1,n
```

```
big=0.0
do j=1,n
  if (ipiv(j)/=1) then
    do k=1,n
      if (ipiv(k)==0) then
        if (abs(a(j,k))>=big) then
          big=abs(a(j,k))
          irow=j
          icol=k
        end if
      else if (ipiv(k)>1) then
        write(45,*) "singular matrix in gaussj - 1", a
      end if
    end do
  end if
end do
ipiv(icol)=ipiv(icol)+1
if (irow/=icol) then
  do l=1,n
    dum=a(irow,l)
    a(irow,l)=a(icol,l)
    a(icol,l)=dum
```

```
  end do
  do l=1,m
    dum=b(irow,l)
    b(irow,l)=b(icol,l)
    b(icol,l)=dum
  end do
end if
indxr(i)=irow
indxc(i)=icol
if (a(icol,icol)==0.0) write(45,*) "singular matrix in gaussj - 1", a
pivinv=1.0/a(icol,icol)
a(icol,icol)=1.0
do l=1,n
  a(icol,l)=a(icol,l)*pivinv
end do
do l=1,m
  b(icol,l)=b(icol,l)*pivinv
end do
do ll=1,n
  if (ll/=icol) then
    dum=a(ll,icol)
    a(ll,icol)=0.0
```

```
      do l=1,n
        a(ll,l)=a(ll,l)-a(icol,l)*dum
      end do
      do l=1,m
        b(ll,l)=b(ll,l)-b(icol,l)*dum
      end do
    end if
  end do
end do
do l=n,1,-1
  if (indxr(l)/=indxc(l)) then
    do k=1,n
      dum=a(k,indxr(l))
      a(k,indxr(l))=a(k,indxc(l))
      a(k,indxc(l))=dum
    end do
  end if
end do

END SUBROUTINE gaussj
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
END MODULE
```

A.2 – FORTRAN 95 CODE FOR MULTIPARTICLE ADHESIVE DYNAMICS SIMULATION

```
PROGRAM MAD

!Modules that contain relevant subprograms for use
USE omp_lib
USE External_Forces
USE cdl_biem

IMPLICIT NONE
integer,parameter::nmesh2=4,&                !describes mesh
        nsurf=1,&                                !# of particles
        nel=360,&                                !# of elements/particle
        nnode=1442,&                             !# of nodes/particle
        saveskip2=100,&                          !saves coordinates every 10^3
time steps
        maxbond=500,&                            !maximum # of bonds formed
between platelet and wall
        ReceptorNodes = 3057246          !Number of sLex ligands on the CTM
surface
double precision,parameter::Gdot=1000.0,&  !shear rate
        PI=3.14159265358979,&
        dens=1.d-6,&                             !density of fluid in microgram/cu. micron
        dt0=1.d-7,&                  !initial time step
        dt1=1.d-8,&                  !time step
        dt2=1.d-9,&                  !time step
        dt3=1.d-10,&                             !time step
        dt4=1.d-11,&                             !time step
        dt5=1.d-12                   !time step
character(*),PARAMETER::suffix="0", mesh = "4"

integer::step,&                  !looping variable:counts no of time steps from 0 to Nt
      is,&                               !counter for particle number
      i, iel,&
      j,&
      counter,&                          !no of iterations for minimizing error, limit is
maxiter
    k,&
    savecount2,&     !counter for no. of iterations from 0-saveskip2
    record,&         !counter for no. of outputs tabulated
    repeat
```

```
!indices of nodes for each element in a unit sphere
integer,dimension(nel,9)::connectu

!indices of nodes for all interacting particles
integer,dimension(nsurf*nel,9)::connect

double precision:: ppeps,tmp
double precision::dt,&          !time step
        dtold,&                 !old time step
        timenew,&       !The current time for platelet motion
        z,&
        signchange,&  !for use in allocating positions for receptor nodes
        minz,&                  !minimum z-axis node coordinate
        maxz            !maximum z-axis node coordinate

double precision,dimension(3)::gravi,&
        xtemp,&                 !temp variable for cartesian coordinates
        center          !of element centroid

double precision,dimension(nsurf)::areas,& !surface area of particle
        density,&       !cell density
        vol             !volume of the cell

double precision, dimension(nsurf,3)::sizes, &          !radius of cell
                                        ceng            !center of gravity
double precision,dimension(nsurf*nel)::ara   !area of an element
double precision,dimension(nsurf,3)::InitX,&!Initial position of cell center
        angles  !angles made by cell wrt x, y, z axis

double precision,dimension(nsurf,6)::extf,&             !external force
        RBMs
double precision,dimension(nsurf*nel,3)::centre !element centroid

double precision, dimension(nsurf)::lowest !clearance between platelet and surface

double precision::  eps         !shortest distance between surface and plane

double precision,dimension(nsurf*nel,4)::rnorml !unit normal for all nel

!coordinates of the nodes of a unit cell
double precision,dimension(nnode,3)::vcdu

!true coordinates for every node of all interacting cells
double precision,dimension(nsurf*nnode,3)::vcd
```

```
double precision, dimension(3*nsurf*nel)::phi

!!!!!!!!!!!!!!!!! VARIABLES FOR RECEPTOR-LIGAND BOND FORMATION
AND DISSOCIATION !!!!!!!!!!!!!!!!!
integer:: ibond !counter for total number of bonds existing in the system at that
instantaneous time

!Number of receptors per element of particle
integer,dimension(nel)::ReceptorsperEle

!Max number of receptors per element of particle
integer::maxReceptorsperEle

!Number of bonds formed in an element
integer,dimension(nsurf*nel)::elebonds

!number of old bonds between each surface
integer,dimension(nsurf,nsurf+1)::nbondold

!tracks the surfaces involved for each existing bond
integer,dimension(maxbond,2)::bind

!the number of the element on each surface at which the bond in question is formed
integer, dimension(nsurf,maxbond)::oldeleno

double precision,dimension(maxbond,7)::bcoordold !coordinates of the endpoints of
every
                    !bond existing at that instantaneous time !7 is the bond status NG=0 or
IG=1

double precision, dimension(nsurf*ReceptorNodes,3):: rnodevcd !coordinates of
receptor locations on platelet surface

double precision, dimension(maxbond)::bondstarttime,bondstartold !the real time
when a bond is formed

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
          integer, dimension(8):: values
          character(8):: date
          character(10):: time

!!!!!!!!!!!!!!!!!!!!!!!!!VARIABLES FOR MANUAL E-SELECTIN LIGAND
DENSITY!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
integer, parameter:: LigandDensity = 2626
        !Number of E-selectin ligands per square micron on plane surface
integer, parameter:: ParticleDensity = 1341                                    !Number of
ligands per square micron on (Colo205) particle surface
double precision:: ligandratio = LigandDensity/ParticleDensity

!!!!!!!!!!!!!!!!!!!!!!!!!!VARIABLES FOR MEASURING SUBROUTINE RUN
TIMES!!!!!!!!!!!!!!!!!!!!!!!!!!!!
real, dimension(2):: setuptarray, bondtarray1, bondtarray2, hydrotarray, totarray
real:: setupresult, bondresult1, bondresult2, tempbondtime, bondtime, hydro1, hydro2,
hydrotime, hydroresult, totresult
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!PROGRAM EXECUTION
BEGINS!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!call Random_Seed()
dt=dt0

! variable initialization
!simulating a 2mer CTM
! radii of particle in the x = 1, y = 2, z = 3 directions
sizes(:,1)=6.9590
sizes(:,2)=6.9590
sizes(:,3)=6.9590

density=1.05d-6;        gravi(1)=0.0;   gravi(2)=0.0;   gravi(3)=-9806650

!Initializing the location of particle center
InitX(1,1)= 0.0
InitX(1,2)= 0.0
InitX(1,3)= 7.8901

!input mesh coordinates and connectivity rules
open(unit=9,file="/home/fs01/kja63/ReceptorDistribution/4mer235R1Coords-
1um.dat",action="READ", form="FORMATTED")
do i=1,nnode
  read(9,*) (vcdu(i,j),j=1,3)
  !2000 format (3F9.6)
end do
open(unit=10,file="/home/fs01/kja63/ReceptorDistribution/4mer235R1Connect-
1um.dat",action="READ")
do i=1,nel
  read(10,*) (connectu(i,j),j=1,9)
end do
close(10)
```

```
!check to see if scaling works
!open(unit=97,file="meshCheck.txt",action="WRITE",status="REPLACE")
!do i=1,nnode
!         write(97,*) (vcdu(i,j),j=1,3)
!end do
!close(97)

!check connect file
!open(unit=96,file="connectCheck.txt",action="WRITE",status="REPLACE")
!do i=1,nel
!         write(96,*) (connectu(i,j),j=1,9)
!end do
!close(96)

!Input platelet surface area that corresponds to each node for a meshed surface
!open(unit=9,file="/home/fs01/ww274/Weiwei/Rolling/Nodeequivarea"//mesh//".dat",
action="READ")
!do i = 1,nnode
!         read(9,*) nodearea(i)
!end do
!close(9)

!input number of sLex present in each element
open(unit=8,file="/home/fs01/kja63/ReceptorDistribution/4mer235-
7um_ReceptorDistributionperEle.dat",action="READ")
!,status="OLD", position="REWIND")
do i = 1, nel
         read(8,*) ReceptorsperEle(i)
end do
close(8)


!!!!!!! OPENING OUTPUT FILES !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! trajectory file
open(unit=11,file="CTMtrans"//suffix//".txt",status="REPLACE")
!!!!!!!!!!!!!!!! angle file
open(unit=12,file="CTMangles"//suffix//".txt",status="REPLACE")
! time, since save frequency is stochastic
!open(unit=17,file="CTMtime"//suffix//".txt",status="REPLACE")
!!!!!! lowest point on platelet
open(unit=20,file="CTMlowpoint"//suffix//".txt",status="REPLACE")
!!!!!! new check: distace between surface and lowest node coord
!open(unit=48,file="CTMdistance"//suffix//".txt",status="REPLACE")
!!!!!!!!!!!!!!timestep change
```

```
open(unit=21,file="CTMtimestep"//suffix//".txt",status="REPLACE")
!!!!!!time for platelet motion
open(unit=22,file="CTMrealtime"//suffix//".txt",status="REPLACE")
open(unit=45, file="CTMinfo"//suffix//".txt",status="REPLACE")
!!!! final bond coordinates
open(unit=15,file="CTMbondcoord"//suffix//".txt")
!!!!!!!!! forces on each bond
open(unit=25,file="CTMbondforces"//suffix//".txt",status="REPLACE")
!!!!!!!!! torques on each bond
!open(unit=26,file="CTMbondtorques"//suffix//".txt",status="REPLACE")
!node and time step at which bond is formed
open(unit=23,file="CTMbformed"//suffix//".txt",status="REPLACE")
!node and time step at which bond is broken
open(unit=24,file="CTMbbroken"//suffix//".txt",status="REPLACE")
!Collision info file
!open(unit=37,file="CTMcollision"//suffix//".txt",status="REPLACE")
!Contact time
!open(unit=40,file="CTMcontacttime"//suffix//".txt",status="REPLACE")
!Tracks no of iterations that are reqd for convergence
open(unit=34,file="CTMcounter"//suffix//".txt",status="REPLACE")
!External forces acting on the platelet
open(unit=35, file="CTMextforces"//suffix//".txt",status="REPLACE")
!record the life span of each bond
open(unit=46, file="CTMbondlifespan"//suffix//".txt",status="REPLACE")
open(unit=47, file="CTMnumbondexist"//suffix//".txt",status="REPLACE")
!Checks bond probabilities
!open(unit=95, file="CTMbondprob"//suffix//".txt",status="REPLACE")
!Secondary check of existing bonds and when breakage loop is run
!open(unit=94, file="CTMbbreakcheck"//suffix//".txt",status="REPLACE")
!Coordinates for manually placed E-selectin receptors
!open(unit=93, file="Eselectincoords"//suffix//".txt",status="REPLACE")
!Cumulative simulation time run for hydrodynamics vs. bond formation/breakage
open(unit=92, file="Runtime"//suffix//".txt",status="REPLACE")
!Checks output of parameters for on rate
!open(unit=91, file="surfvcheck"//suffix//".txt",status="REPLACE")
!Checks requirements for bond formation/breakage evaluation
!open(unit=90, file="centrecheck"//suffix//".txt",status="REPLACE")
!Checks cell surface receptor coordinates
!open(unit=89, file="Receptorcoords"//suffix//".txt",status="REPLACE")
open(unit=88, file="repulsiveForceCheck"//suffix//".txt",status="REPLACE")
!Checks if reduction variables are summed correctly
!open(unit=37, file="reductioncheck"//suffix//".txt",status="REPLACE")
!Checks what step the omp directives work until
!open(unit=40, file="parallelcheck"//suffix//".txt",status="REPLACE")
```

```
!Checks thread assignments for running code in parallel
!open(unit=87, file="threadcheck"//suffix//".txt",status="REPLACE")
!Checks modified avgtotsurfv array
!open(unit=86, file="stepCountCheck"//suffix//".txt",status="REPLACE")
!Checks if timestep is properly lowered after a bond is broken
open(unit=85, file="timestepcheck"//suffix//".txt",status="REPLACE")
!Checks if InitX is correctly calculated from ceng
open(unit=84,file="testInitX"//suffix//".txt",status="REPLACE")
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

call date_and_time(VALUES = values)
write(45,*) values(3), ' day of ',values(2),' month and year ',values(1),'. Time
',values(5),':',values(6)
write(45,*) "distance from surface= ",InitX(1,3)!, "and", InitX(2,3)
!write(45,*) "initial X center-center separation distance =", InitX(2,1)
!write(45,*) "initial Y center-center separation distance =", InitX(2,2)
write(45,*) "shear rate = ",Gdot
write(45,*) 'lambdasLex-Eselectin =',lambdaGPvWFGP

do repeat = 1,1
write(45,*) "repeat", repeat

do is=1,nsurf
  do i=1,nnode
    do j=1,3
      vcd(i+(is-1)*nnode,j)=vcdu(i,j)*sizes(is,j)
    end do
  end do
  do i=1,nel
    do j=1,9
      connect(i+(is-1)*nel,j)=connectu(i,j)+(is-1)*nnode
    end do
  end do
end do

! set InitX(1,3) = lowest point in particle + pbump + zmax
call calcceng(vcd, connect, nel, nsurf, ceng)
minz = minval(vcd(:,3))
InitX(1,3)=(ceng(1,3)-minz)+0.35+(0.2705544+(0.02*1.1))

write(84,*) ceng
write(84,*) maxval(vcd(:,3)),minval(vcd(:,3))
write(84,*) InitX(1,3)
close(84)
```

```
! now correct the node coordinates for any bias in ceng calc
! the center of gravity of an element cell will be different from that of a sphere
! ceng much be corrected to the initial value desired. This can be done by
! shifting vcd values by the difference between the two centers (desired and actual).
! If ceng is behind InitX, the vcds will be moved forward
! if ceng is in front of Init X the vcds will be moved backwards.

do is=1,nsurf
  do i=1,nnode
    do j=1,3
      vcd(i+(is-1)*nnode,j)=vcd(i+(is-1)*nnode,j)+InitX(is,j)-ceng(is,j)
    end do
   end do
end do
ceng=InitX

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Initializing arrays
nbondold=0
angles=0.0
elebonds=0
bcoordold = 0.0
oldeleno = 0
bondstarttime = 0.0
bondstartold = 0.0
!nodecontactextent = 0.0
!contactarea = 0.0
!contacttime = 0.0; maxcontactarea = 0.0; timeintegralcontactarea=0.0

!Initializing variables
savecount2=0; record=0;        step=0;ibond=0;  timenew=0;           counter=1;
            ppeps = 2.0;
tempbondtime=0; bondtime=0; hydro1=0; hydro2=0; hydrotime=0;
maxReceptorsperEle = maxval(ReceptorsperEle(1:nel-1));

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
call init_random_seed()

!Determine total run time for setup
call dtime(setuptarray,setupresult)
write(92,*) setupresult

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! BEGIN TIME LOOP !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
do while (ceng(1,1) < 500000.0 .and. ceng(1,1) > -10.0)
if (savecount2==saveskip2) savecount2=0
savecount2=savecount2+1
if (savecount2==saveskip2) record=1
step=step+1

! particle/mesh properties <---------------------------------------
call particle_mesh_properties(nsurf, nel, vcd, connect, centre, rnorml, areas, ara, vol,
ceng)

! external forces and torques on each particle <--------------------
extf = 0.0

!Gravity forces
do is=1,nsurf
  do i=1,3
    extf(is,i)=-vol(is)*(density(is)-dens)*gravi(i)
  end do
end do

!To determine the magnitude of the short-range repulsive force acting at the tips of the
! surface roughness layers of two surfaces
call Repulsive_force_between_particle_and_wall(step, nsurf, nnode, vcd, extf, ceng,
lowest,eps)

!Run time check start
call dtime(bondtarray1,bondresult1)

call Bond_formation_breakage(maxbond, nsurf, nel, step, dt, timenew,
ReceptorsperEle, centre, connect, vcd, &
                    bcoordold, elebonds, bind, oldeleno, nbondold, ibond,
Gdot,bondstarttime,bondstartold, record, RBMs, ceng, &
                    ligandratio, maxReceptorsperEle)

call Bond_forces(maxbond, ibond, bcoordold, bind, ceng, nsurf, record, timenew, extf,
oldeleno)

!Run time check end
call dtime(bondtarray2, bondresult2)

call main_cdlbiem(Gdot, nsurf, nnode, nel, step, vcd, connect, ceng, areas, ara, centre,
rnorml, sizes, extf, phi, RBMs,&
                    counter)
```

```
!To determine the correct time step
dtold = dt
if (counter<75) then
        dt = dt0
        else if (counter<150) then
        dt = dt1
        else if (counter<250) then
        dt = dt2
        else if (counter<350) then
        dt = dt3
        else if (counter<500) then
        dt = dt4
        else
        dt = dt5
end if
!Lower timestep more if bond event occurred on previous step
if (bondformed .or. bondbroke) then
        dt = dt/100
        write(85,*) NINT(dt0/dt), step
end if

if (step == 1) write(21,*) NINT(dt0/dt), step
if (dt /= dtold) write(21,*) NINT(dt0/dt), step

timenew = dt+timenew
!if (lowest < lambda) then
!       contacttime = contacttime+dt
!       timeintegralcontactarea = timeintegralcontactarea+dt*contactarea
!       if (maxcontactarea(1)< contactarea(1)) maxcontactarea(1) = contactarea(1)
!       if (maxcontactarea(2)< contactarea(2)) maxcontactarea(2) = contactarea(2)

!end if

if (counter>maxiter) then
        write(45,*) 'step: ',step
        write(45,*) 'lowest, extf, vel = RBMs(1,i)'
        write(45,*) lowest
        write(45,*) extf
        do i = 1,6
                write(45,*) RBMs(1,i)
        end do
        write(45,*) ''
end if
```

```
! update particle positions if not directly after breakage
write(34,*) counter

! update particle positions if not directly after breakage
if (record==1) then
  write(20,*) step, lowest
  write(22,*) step, timenew
end if
do is=1,nsurf
  if (record==1) then
    write(11,*) (ceng(is,j),j=1,3)
    write(12,*) (angles(is,i),i=1,3)
    if (is==nsurf)  record=0
  end if
end do

!**********************************************************************
**********************
! Algorithm to curtail errors by specifying a cutoff lower limit for recognizable motion
! The diameter of an atomic nucleus is 10^-15 m and the atomic diameter is 10^-11 m.
!For distances less than 1 picomicron, then distance is treated as zero.

do is = 1, nsurf
          do j = 1,3
                    if (abs(dt*RBMs(is,j)) < 1.d-6) RBMs(is,j) = 0
          end do
end do
!**********************************************************************
*********************

do is=1,nsurf
  do i=1+(is-1)*nnode,is*nnode
    do j=1,3
      xtemp(j)=vcd(i,j)
      center(j)=ceng(is,j)
    end do
    do j=1,3
      call rotate(xtemp,center,j,dt*RBMs(is,j+3))
    end do
    do j=1,3
      vcd(i,j)=xtemp(j)+dt*RBMs(is,j)
    end do
  end do
end do
```

```
do i=1,ibond
  if (bind(i,2)<=nsurf) then
    do j=1,3
      xtemp(j)=bcoordold(i,j)
      center(j)=ceng(bind(i,1),j)
    end do
    do j=1,3
      call rotate(xtemp,center,j,dt*RBMs(bind(i,1),j+3))
    end do
    do j=1,3
      bcoordold(i,j)=xtemp(j)+dt*RBMs(bind(i,1),j)
      xtemp(j)=bcoordold(i,j+3)
      center(j)=ceng(bind(i,2),j)
    end do
    do j=1,3
      call rotate(xtemp,center,j,dt*RBMs(bind(i,2),j+3))
    end do
    do j=1,3
      bcoordold(i,j+3)=xtemp(j)+dt*RBMs(bind(i,2),j)
    end do
  else
    do j=1,3
      xtemp(j)=bcoordold(i,j)
      center(j)=ceng(bind(i,1),j)
    end do
    do j=1,3
      call rotate(xtemp,center,j,dt*RBMs(bind(i,1),j+3))
    end do
    do j=1,3
      bcoordold(i,j)=xtemp(j)+dt*RBMs(bind(i,1),j)
    end do
  end if
end do

do is=1,nsurf
  do i=1,3
    ceng(is,i)=ceng(is,i)+dt*RBMs(is,i)
    angles(is,i)=angles(is,i)+dt*RBMs(is,i+3)
  end do
end do

!Calculate and output run time
call dtime(hydrotarray,hydroresult)
```

```
call etime(totarray,totresult)
tempbondtime = bondresult2;
bondtime = bondtime + tempbondtime;
hydro1 = hydro1 + bondresult1;
hydro2 = hydro2 + hydroresult;
hydrotime = hydro1 + hydro2;
write(92,*) hydrotime, bondtime, totresult



call flush

end do


write(45,*) 'repeat no :',repeat
do is = 1, nsurf
        write (45,*) (ceng(is,i),i=1,3)
end do

!write(45,*) 'Job ended on ',datdtime

! Release all contents from buffer
call flush()

!write to file the current data and time
call date_and_time(VALUES = values)
write(45,*) values(3), ' day of ',values(2),' month and year ',values(1),'. Time
',values(5),':',values(6)


end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!! END TIME LOOP !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

contains

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine rotate(x,center,dir,angle)
integer, INTENT(IN)::dir
integer,dimension(2)::e12
double precision, INTENT(IN)::angle
double precision::rrot,theta
```

```
double precision,dimension(3),INTENT(IN)::center
double precision, dimension(3),INTENT(OUT)::x

if (dir==1) then
  e12(1)=2;      e12(2)=3
elseif (dir==2) then
  e12(1)=3;      e12(2)=1
else
  e12(1)=1;      e12(2)=2
end if

x(e12(1))=x(e12(1))-center(e12(1))
x(e12(2))=x(e12(2))-center(e12(2))
rrot=sqrt(x(e12(1))**2+x(e12(2))**2)
theta=atan2(x(e12(2)),x(e12(1)))+angle
x(e12(1))=rrot*cos(theta)+center(e12(1))
x(e12(2))=rrot*sin(theta)+center(e12(2))

end subroutine rotate

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     SUBROUTINE init_random_seed()
       INTEGER :: i, n, clock
       INTEGER, DIMENSION(:), ALLOCATABLE :: seed

       CALL RANDOM_SEED(size = n)
       ALLOCATE(seed(n))

       CALL SYSTEM_CLOCK(COUNT=clock)

       seed = clock + 37 * (/ (i - 1, i = 1, n) /)
       CALL RANDOM_SEED(PUT = seed)

       DEALLOCATE(seed)
     END SUBROUTINE


END PROGRAM MAD
```

A.3 – FORTRAN 95 CODE FOR EXTERNAL FORCES CALCULATION FOR MULTIPARTICLE ADHESIVE DYNAMICS SIMULATION

```
MODULE External_Forces

USE omp_lib

IMPLICIT NONE
        double precision, PARAMETER:: lambda=0.02,&    !equilibrium bond length for bimolecular bond (lambda(0.02)+bump)
          lambdaGPvWFGP = 0.02,&           !Equilibrium bond length for trimolecular bond (Platelet-vWF-platelet; not present for Colo-205)
          sigmatensile=1.0d7,&              !spring constant of molecular bond
          sigmacompressed=1.0d7,& !spring constant of molecular bond
          kf0=4.0,&                         !intrinsic rate of bond formation (from Chang and Hammer 2000)
          gamma=2.0d-5,&! 1.8d-5,& !        !Bell model parameter (ro in eqn) aka reactive compliance
          kr0_ng=0.44,&! 5.47,&! 3.21,& !   !unstressed rate of dissociation
          kr0_ig=0.44,&
          kr0=0.44,&          !reverse rate of dissociation (from Chang and Hammer 2000)
          bump=0.2705544,&                  !roughness of surface (steric layer~271 nm)
          pbump=0.35,&                      !roughness of plane (Endothelial Glycocalyx Layer)
          kT=1.3806488d-2*298,&         !Boltzmann's constant times temperature
          tau=1666.6667,&                   !tau in repulsive force equation (t^-1)
          frep=4.0d15,&                     !Fo in repulsive force equation

          ! catch-slip parameters, see Auton et al. Biophysical J. 2010
          K0= 0.3,&    !force free NG - IG equilibrium constant
          Sigma= 0.45d-3  !0.45nm

        logical :: bondformed, bondbroke

        !Variable related to running code in parallel
        integer, PARAMETER :: nthreads=5 !number of threads to be run in parallel
        integer :: TID  !thread ID number

CONTAINS

SUBROUTINE Repulsive_force_between_particles(nnode, vcd, connectu,
```

```
record,issphere, nsurf, nel, ceng, sizes, nodearea, extf, &
                ppeps, contactarea, nodecontact, nodecontactextent, dt, dt0)
!variables for platelet-platelet repulsive force calculations

        integer, INTENT(IN) :: record, nsurf, nnode, nel
        integer, dimension(nsurf, nnode), INTENT(INOUT)::nodecontact
        double precision, dimension(nsurf, nnode),
INTENT(INOUT)::nodecontactextent
        logical, INTENT(IN):: issphere
        double precision, INTENT(IN):: dt, dt0
        double precision,dimension(nsurf,3), INTENT(IN):: ceng, sizes
        double precision, dimension(nnode), INTENT(IN)::nodearea
        double precision, dimension(nsurf,1:6), INTENT(INOUT) :: extf
        double precision,dimension(nsurf*nnode,3), INTENT(IN):: vcd
        integer, dimension(nel,9),INTENT(IN)::connectu
        double precision, INTENT(OUT):: ppeps      !The closest distance of approach
between two platelets
        integer::vcdlowm,&    !The closest node point on platelet 1
                 vcdlown,&    !The closest node point on platelet 2
                 i, j, k, l, is, js, node, iel          !counters
        double precision:: eps, &       !shortest distance between two particle surfaces
                           dist         !Distance between two nodes of the closest
quadrants
        double precision, dimension(2), INTENT(OUT)::contactarea
        double precision, dimension(3) :: frough
     logical, dimension(nnode)::considered1, considered2 !Logical vectors that
remember which nodes
                                                        !have been taken into account during
contact area calc

        frough(:) = 0.0
        contactarea(:) = 0.0
        considered1 = .false.
        considered2 = .false.

        !if (issphere) then
        !Repulsive force between two spherical particles
        !do is=1,nsurf
        !  do js=is+1,nsurf
        !     eps=(sqrt((ceng(is,1)-ceng(js,1))**2+(ceng(is,2)-ceng(js,2))**2+&
        !        (ceng(is,3)-ceng(js,3))**2)-sizes(is,3)-sizes(js,3)-2.*bump)
        !         do i=1,3
        !                 frough(i)=frep*exp(-tau*eps)/(1.0-exp(-tau*eps))*&
        !                  (ceng(is,i)-ceng(js,i))/(eps+sizes(is,3)+sizes(js,3)+bump)
```

```
!                   extf(js,i)=extf(js,i)+frough(i)/2
!                   extf(is,i)=extf(is,i)-frough(i)/2
!           end do
!  end do
!end do
!ppeps = eps+2*bump
!else
            do i = 1,nnode !first platelet
                  do j = 1,nnode !second platelet
                        dist = sqrt((vcd(i,1)-vcd(nnode+j,1))**2+(vcd(i,2)-
vcd(nnode+j,2))**2+(vcd(i,3)-vcd(nnode+j,3))**2)-2*bump
                        if (i==1.AND.j==1) then
                              eps = dist
                              vcdlowm = i
                              vcdlown = j
                        elseif (eps>dist) then
                              eps = dist
                              vcdlowm =  i
                              vcdlown = j
                        end if
                        if (dist < lambdaGPvWFGP - 2*bump) then
                              if(.NOT.considered1(i)) then

        contactarea(1)=contactarea(1)+nodearea(i)
                                    considered1(i)=.TRUE.
                              end if
                              if (.NOT.considered2(j)) then
                                    contactarea(2) =
contactarea(2)+nodearea(j)
                                    considered2(j)=.TRUE.
                              end if
                              !do is = 1, nsurf
                              !     if (is==1) then
                              !           node = i
                              !     elseif (is==2) then
                              !           node = j
                              !     end if
                              nodecontact(1,i) = 1 !replace i with node when
do loop used
                              nodecontactextent(1,i) = nodecontactextent(1,i)
+ dt/dt0 !Time spent in contact for each node
                              !end do !is
                        end if !(dist < lambdaGPvWFGP - 2*bump)
                  end do !j
```

```
                end do !i
                ppeps = eps+2*bump
                if (eps<=0.01) eps = 0.01 ! If eps<=0.01, it causes the frough
calculation to blow up before cdl-biem calculation fails
                do i=1,3
                        frough(i)=frep*exp(-tau*eps)/(1.0-exp(-
tau*eps))*(vcd(vcdlowm,i)-vcd(nnode+vcdlown,i))/ppeps
                        !Frough is directed towards platelet 1. However, forces act in
opposite direction of translation. Hence
                        extf(1,i)=extf(1,i)-frough(i) ! removed divide by 2
                        extf(2,i)=extf(2,i)+frough(i) ! removed the divide by 2
                end do
        !end if
END SUBROUTINE Repulsive_force_between_particles


SUBROUTINE Repulsive_force_between_particle_and_wall(step, nsurf, nnode, vcd,
extf, ceng, lowest,eps)
!This subroutine calculates the magnitude of the repulsive force acting between the
particle
! and the planar surface, for each particle

        integer, INTENT(IN) :: step, nsurf, nnode
        double precision,dimension(nsurf*nnode,3), INTENT(IN):: vcd
        double precision, dimension(nsurf,6), INTENT(INOUT) :: extf
        double precision,dimension(nsurf,3), INTENT(IN):: ceng
        double precision, dimension(nsurf),INTENT(OUT) ::lowest
        double precision, INTENT(OUT) ::  eps       !shortest distance between surface
and plane
        double precision, dimension(3) :: frough, trough, er
        integer is, i, lownode, j
        lowest = ceng(:,3) !correctly executes

        !Determining the point on the platelet(s) that is closest to the surface
        do is = 1,nsurf
                frough(:) = 0.0; trough(:) = 0.0
!               lowest(is) = minval(vcd((is-1)*nnode+1:is*nnode,3))
                do i = (is-1)*nnode+1,is*nnode
                        if (lowest(is)>vcd(i,3)) then
                                lowest(is) = vcd(i,3)
                                lownode = i
                        end if
                end do
                eps=(lowest(is)-bump-pbump)
```

```
                if (eps<=0.01) eps = 0.01 ! If eps<=0.01, it causes the frough
calculation to blow up before the cdl-biem calculation fails
                frough(3)=frep*exp(-tau*eps)/(1.0-exp(-tau*eps))
                extf(is,3)=extf(is,3)-frough(3)

                !Calculating the torque on the platelet due to repulsion
                !Torque Arm length
              er(1)=vcd(lownode,1)-ceng(is,1)
              er(2)=vcd(lownode,2)-ceng(is,2)

                !Calculated by cross product determinant formula
            trough(1)= er(2)*frough(3)
              trough(2)= -er(1)*frough(3)
              do j=1,2
                  extf(is,j+3)=extf(is,j+3)+trough(j)
              end do
          end do

          !Checking distance of closest point to surface
          !write(48,*) step, eps

          !Checking repulsive force output
          if (eps<=0.03) write(88,*) step, eps, frough(3), extf(1,3)

END SUBROUTINE Repulsive_force_between_particle_and_wall



SUBROUTINE Bond_formation_breakage(maxbond, nsurf, nel, step, dt, timenew,
elebondsmax, centre, connect, vcd, &
                bcoordold, elebonds, bind, oldeleno, nbondold, ibond,
Gdot,bondstarttime, bondstartold,record, RBMs, ceng, &
                ligandratio, maxReceptorsperEle)
!This subroutine determines the event of the formation and breakage of bond(s)
between the particle
! and the surface, and the location of the bond formation/breakage event. This is
determined for each
! particle in the system. Receptors are randomly placed within the elements. The
number of receptors that
! can form within an element is based on the area of the element in proportion to the
total surface area of
!the platelet and is read from a file in the main program.
!New model: Location of end of receptor is on the surface and spans the glycocalyx
```

```
        integer, INTENT(IN):: nsurf, nel, step, maxbond,record, maxReceptorsperEle
        integer, INTENT(OUT)::ibond
        integer, dimension(nel), INTENT(IN)::elebondsmax
        integer,dimension(nsurf*nel), INTENT(INOUT)::elebonds
        integer,dimension(nsurf*nel,9), INTENT(IN) ::connect
        integer,dimension(:,:), INTENT(OUT)::bind
        double precision, INTENT(IN) :: dt, timenew, Gdot, ligandratio
        double precision, dimension(nel, 3), INTENT(IN) :: centre
        double precision,dimension(:, :), INTENT(IN):: vcd, RBMs, ceng
        integer, dimension(nsurf,nsurf+1),INTENT(INOUT)::nbondold
        integer, dimension(:,:), INTENT(INOUT)::oldeleno
        double precision,dimension(maxbond,7), INTENT(INOUT)::bcoordold
        double precision,dimension(3)::bforce, fbond1, surfv, relposi
        double precision,dimension(:)::bondstarttime, bondstartold
        integer :: is, iel,&       !counters for nsurf and nel
              nbond,&          !no of bonds existing between two surfaces
              oldind,& !counter for bond number in bcoordold
              lastbond,&       !number of unattached bonds per element before
attachment loop begins
              bondsperele,&    !counter for bond formation at each node
                 i, j, k, ielcheck, &              !counters
                 loccount, &    !counter for surfv averaging
                 !when arrays are allocated, their elements are not necessarily given a
value of zero
                 !therefore, need to set counters to determine how many elements should
be averaged
                 ng_ig, &   !ng/ig status with ng=0,ig=1
                 stepCount  !modulus of current step, used in averaging surfv; ex:
stepCount = 1 for step 1, 10001, 20001, etc.
        double precision :: zmax = (lambda*1.1)+bump, & !maximum distance of
separation over which bonding can occur (lambda + 10% lambda)
                 z, &    !height of receptor from the surface
                 kf, &   !bond association constant
                 prob, &
                 kr, &   !bond dissociation constant
           tmp_length, rkf0,Kng_ig, &
                       magnitude, &          !distance between individual sLex and E-
selectin pair
                       breakmagnitude, &     !distance between individual bonded
sLex and E-selectin pair
                       tempbreakmagnitude
                       !avglocsurfv, &
                       !avgtotsurfv
   double precision, dimension(3)::x !Stores the coordinates of the location of a
```

```
receptor on an element
        double precision,dimension(maxbond,7)::bcoord !coordinates of the endpoints
of every bond existing at that instantaneous time, 7 is NG/IG status
        integer, dimension(nsurf,maxbond)::eleno !the number of the node at which
the bond in question is formed
        !integer, INTENT(IN):: windowligands        !number of E-selectin molecules
in dynamic window
        !integer, dimension(windowligands):: availability     !array to signify whether
E-selectin molecule is currently bound (0=not bound, 1=bound)
        double precision, dimension(maxReceptorsperEle):: locsurfv          !array
containing all surfv values for current element
        double precision, dimension(nsurf,nel):: avglocsurfv            !array containing
average surfv values for all elements in current timestep
        double precision, dimension(nsurf,10000):: avgtotsurfv       !array containing
average surfv values for each timestep (second value the number of steps being
averaged)
        logical, dimension(nsurf,nel):: totcheck        !array that determines which
elements have locsurfv values

 ibond = 0;     oldind = 0;      bondformed = .false.; bondbroke = .false.
 !availability = 0
 !kcheck = 0
 ielcheck = 0
 totcheck(:,:) = .false.
 avgtotsurfv(:,:) = 0.0
 locsurfv(:) = 0.0
 avglocsurfv(:,:) = 0.0
 !determine stepCount based on current step
 stepCount = mod(step,10000)
 if (stepCount == 0) then
        stepCount = 10000     !want every 10000th step to be the last index in array
 end if
 !write(86,*) stepCount
 !write(86,*) avgtotsurfv

 !if (.not. allocated(avglocsurfv)) then
        !allocate(avglocsurfv(nsurf,nel))
 !end if
 !if (.not. allocated(avgtotsurfv)) then
  !allocate(avgtotsurfv(nsurf,step))
 !end if

 do is=1,nsurf
  nbond=0
```

```
   !check for new bond formation
 !$OMP PARALLEL DEFAULT(SHARED) NUM_THREADS(nthreads)
        !write(40,*) step
 !$OMP DO FIRSTPRIVATE(locsurfv) &
 !$OMP PRIVATE(iel, lastbond, loccount, bondsperele, x, z, i, relposi, surfv, rkf0, kf,
prob, j, timenew, tmp_length) &
 !$OMP REDUCTION(+:nbond,ibond,ielcheck) &
 !$OMP SCHEDULE(DYNAMIC)
   do iel = 1+(is-1)*nel,is*nel
                !Check to see if simulation enters bond formation/breakage loop
                !if (centre(iel,3) < 3) then
                        !write(90,*) step, iel, centre(iel,3)
                !end if
                !TID = omp_get_thread_num()
        if (centre(iel,3)-pbump-bump<zmax+0.216) then !2.0 microns/16 squares
(mesh-8)in a column /2 to get half height of a square
                lastbond =elebondsmax(iel-(is-1)*nel) - elebonds((is-1)*nel+iel)
                        ielcheck = ielcheck + 1
                        !write(37,*) step, TID, ielcheck
                        totcheck(is,iel) = .true.          !current element (is,iel) will have
avglocsurfv value
                        !if (.not. allocated(locsurfv)) then
                        !       allocate(locsurfv(lastbond))
                        !end if
                        loccount = 0
                        do bondsperele = 1,lastbond
                                x = Receptorcoord(iel, vcd, connect)
                                z = x(3) - pbump
                                        do i=1,3
                                          relposi(i) = x(i)-ceng(is,i)
                                        end do
                                        !write(89,*) x
                                        surfv=0.0
                                        surfv(1)=
surfv(1)+RBMs(is,1)+sqrt(relposi(1)**2+relposi(3)**2)*RBMs(is,5)*sin(atan2(relpos
i(3),relposi(1)))
                                        surfv(1)= surfv(1)-
sqrt(relposi(1)**2+relposi(2)**2)*RBMs(is,6)*sin(atan2(relposi(2),relposi(1)))
                                        surfv(2)= surfv(2)+RBMs(is,2)-
sqrt(relposi(2)**2+relposi(3)**2)*RBMs(is,4)*sin(atan2(relposi(3),relposi(2)))
                                        surfv(2)=
surfv(2)+sqrt(relposi(1)**2+relposi(2)**2)*RBMs(is,6)*sin(atan2(relposi(1),relposi(2
)))
                                        surfv(3)= sqrt(surfv(1)**2+surfv(2)**2)
```

```
                                        !parallelization thread check
                                        !if (bondsperele == 1) then
                                        !       write(87,*) step, TID, iel, lastbond
                                        !end if
                                        !averaging surfv
                                        locsurfv(bondsperele) = surfv(3)
                                        loccount = loccount + 1
                                        !if (z<zmax) then
                                        !       write(91,*) step, TID, "iel ", iel, "locsurfv
", locsurfv(bondsperele)
                                        !end if
                                        if (bondsperele == 1) then
                                                avglocsurfv(is,iel) =
locsurfv(bondsperele)
                                        else if (bondsperele <= lastbond) then
                                                avglocsurfv(is,iel) =
sum(locsurfv(1:loccount))/size(locsurfv(1:loccount))
                                                !if (z<zmax) then
                                                !       write(91,*) step, TID, "iel ", iel,
"avglocsurfv ", avglocsurfv(is,iel)
                                                !end if
                                        end if
                                        !!$omp critical
                                        if (iel == 1) then
                                                avgtotsurfv(is,stepCount) =
avglocsurfv(is,iel)
                                        else if (iel <= is*nel) then
                                                avgtotsurfv(is,stepCount) =
sum(pack(avglocsurfv(is,:),avglocsurfv(is,:) /= 0.0))/ielcheck
                                                !if (z<zmax) then
                                                !       write(91,*) TID, "step ", step,
"avgtotsurfv ", avgtotsurfv(is,stepCount)
                                                !end if
                                        end if
                                        !!$omp end critical
                                        !average 10000 iterations of surfv for rkf0
calculation
                                        if (stepCount <= 10000) then
                                                if (stepCount == 1) then
                                                        rkf0 = kf0 *
avgtotsurfv(is,stepCount)
                                                else
                                                        rkf0 = kf0 *
(sum(pack(avgtotsurfv,avgtotsurfv /= 0))/size(pack(avgtotsurfv,avgtotsurfv /= 0.0)))
```

```
                                        end if
                                    else
                                        rkf0 = kf0 *
(sum(avgtotsurfv)/size(avgtotsurfv))
                                    end if

                                    !if (mod(step,100) == 0) then
                                    !       write(86,*) step, TID,
avgtotsurfv(is,stepCount)
                                    !end if
                                !rkf0 = kf0 * surfv(3)
                                !Check to see if surfv averaging works
                                !if (z<zmax) then
                                    !write(91,*) step, z, rkf0
                                !end if

                                if (z<zmax) then
                                    !do k = 1, windowligands             !cycle
through E-selectin molecules to see if any are close to sLex ligand
                                        !magnitude = sqrt((x(1)-
ligandcoords(k,1))**2 + (x(2)-ligandcoords(k,2))**2)
                                        !if (magnitude >= 0*0.02 .and. magnitude
<= 0.1*0.02 .and. availability(k)==0) then
                                            kf=rkf0*exp(sigmatensile*abs(z-
bump-lambda)*(gamma-abs(z-bump-lambda)/2.0)/kT) !Eqn (3) from King and
Hammer, 2001
                                            if (kf <= 0.01) then
                                                kf = 0.01
                                            end if
                                                call
Random_Number(prob)
                                            do while (prob==0.0)
                                                    call
Random_Number(prob)
                                            end do
                                            if ((prob/ligandratio)<1.0-exp(-
kf*dt)) then
                                            !$omp critical
                                            write(95,*), step, dt, z, kf, prob

          write(25, *) kf0, kf
                                                    nbond=nbond+1;
        ibond=ibond+1
                                                    !write(37,*) step, TID,
```

```
nbond, ibond
                                                        do j=1,2

        bcoord(ibond,j)=x(j)

        bcoord(ibond,j+3)=x(j)
                                                        end do
                                                        bcoord(ibond,3)=x(3)
                                                        bcoord(ibond,6)=pbump
                                                        bcoord(ibond,7)=0.0
                                                        bind(ibond,1)=is
                                                        bind(ibond,2)=nsurf+1
                                                        elebonds((is-
1)*nel+iel)=elebonds((is-1)*nel+iel)+1
                                                        eleno(is,nbond)=iel

bondstarttime(ibond)=timenew

          tmp_length=sqrt((bcoord(ibond,1)-bcoord(ibond,4))**2+&

(bcoord(ibond,2)-bcoord(ibond,5))**2+&

(bcoord(ibond,3)-bcoord(ibond,6))**2)
                                                        write(45,*) "bond
formed! element no", iel, "nbond", nbond, "time step", step, "b_length", tmp_length
                                                        write(23,*)iel,step,z
                                                        bondformed=.true.
                                                        !availability(k) = 1
                                                    !$omp end critical
                                                    end if
                                                    EXIT
                                                !end if
                                            !end do
                                        end if
                                end do
                                locsurfv(:) = 0.0
                                !if (allocated(locsurfv)) then
                                !       deallocate(locsurfv)
                                !end if
                        end if
  end do
!$OMP END DO
!$OMP END PARALLEL
        avglocsurfv(:,:) = 0.0
```

```
        !~if (allocated(avglocsurfv)) then
        !~      deallocate(avglocsurfv)
        !~end if

!Check for bond breakage
        !check bond breakage parameters
        !write(94,*) "Step: ", step
        !write(94,*) "# old bonds = ", nbondold

   if (nbondold(is,nsurf+1)>0) then
        do i=1,nbondold(is,nsurf+1)
                        oldind=oldind+1
             z=sqrt((bcoordold(oldind,1)-bcoordold(oldind,4))**2+&
                  (bcoordold(oldind,2)-bcoordold(oldind,5))**2+&
            (bcoordold(oldind,3)-bcoordold(oldind,6))**2)
        Kng_ig = K0*exp(Sigma*sigmatensile*(z-bump-lambda)/kT)

!Determine whether bond status will change between NG and IG.
                        call Random_Number(prob)
                        do while (prob==0.0)
                                call Random_Number(prob)
                        end do

!             kr=1./(1.+kng_ig)*kr0_ng*exp(y_ng*sigmatensile*(z-lambda)/kT)
!             kr=kr+kng_ig/(1.+kng_ig)*kr0_ig*exp(y_ig*sigmatensile*(z-lambda)/kT)
             kr=kr0*exp((gamma*sigmatensile*(z-bump-lambda))/kT)  !parameter from
Smith et al, Biophysical J. 1999

!   if (bcoordold(oldind,7)==0) then
!       if(prob>1.0-exp(-k_ng_ig*dt)) then
!          kr=kr0_ng*exp(y_ng*sigmatensile*(z-lambda)/kT)
!       else
!          bcoordold(oldind,7)=1
!          kr=kr0_ig*exp(y_ig*sigmatensile*(z-lambda)/kT)
!       end if
!    else if (bcoordold(oldind,7)==1) then
!       if(prob>1.0-exp(-k_ig_ng*dt)) then
!          kr=kr0_ig*exp(y_ig*sigmatensile*(z-lambda)/kT)
!       else
!          bcoordold(oldind,7)=0
!          kr=kr0_ng*exp(y_ng*sigmatensile*(z-lambda)/kT)
!       end if
!    end if
```

```
!               if (prob<(1/(1+Kng_ig))) then
!                       kr=kr0_ng*exp(y_ng*sigmatensile*(z-lambda)/kT)
!                       ng_ig=0
!               else
!                       kr=kr0_ig*exp(y_ig*sigmatensile*(z-lambda)/kT)
!                       ng_ig=1
!               end if

!               kr=kr0*exp(gamma*sigmatensile*(z-lambda)/kT)
!                       kr=10*exp(4.0)+kr0*exp(gamma*sigma*abs(z-
lambda)/kT)*exp(sigma*abs(z-lambda)/5000)
!                       kr=kr/(exp(4.0)+exp(sigma*abs(z-lambda)/5000))
                        call Random_Number(prob)
                        do while (prob==0.0)
                                call Random_Number(prob)
                        end do

                        !check bond breakage parameters
                        !write(94,*) "Step: ", step
                        !write(94,*) "Record status = ", record

                        !if (record==1) then
                        !       write(15,*) ibond, bcoord(ibond,1), bcoord(ibond,2),
bcoord(ibond,3), bcoord(ibond,4), bcoord(ibond,5), bcoord(ibond,6)
                        !end if

                if (prob>1.0-exp(-kr*dt)) then
                        nbond=nbond+1
                ibond=ibond+1
                eleno(is,nbond)=oldeleno(is,oldind)
                bind(ibond,1)=is
                bind(ibond,2)=nsurf+1
                do j=1,6
                                bcoord(ibond,j)=bcoordold(oldind,j)
                end do
                bondstarttime(ibond)=bondstartold(oldind)
                        else
                                elebonds((is-1)*nel+oldeleno(is,oldind))=elebonds((is-
1)*nel+oldeleno(is,oldind))-1
                                        write(45,*) "BREAKAGE OCCURRED!!!!!!!!",
"element no", oldeleno(is, oldind), "time step", step, "b_leng", z
```

```
        write(24,*)oldeleno(is,oldind),step,z
                                        bondbroke = .true.

                                        do j=1,3
                                                fbond1(j)=sigmatensile*(z-
lambda)*(bcoordold(oldind,j+3)-bcoordold(oldind,j))/z
                                         end do
                                         write(46,*) timenew-bondstartold(oldind)

                                        bforce(1)=oldeleno(is,oldind)
                                        bforce(2) = (z-lambda)/abs(z-
lambda)*sqrt(fbond1(1)**2+fbond1(2)**2+fbond1(3)**2)
                                        bforce(3) = timenew
                                                write(45,*) (bforce(j),j=1,3)
                end if
                   end do
   end if
   nbondold(is,nsurf+1)=nbond
   do i=1,nbond
         oldeleno(is,i)=eleno(is,i)
   end do
  end do !is = 1, nsurf
  bcoordold = bcoord
  bondstartold = bondstarttime
  if (record==1) then
   write(47,*)nbond
  end if

END SUBROUTINE Bond_formation_breakage

SUBROUTINE Bond_forces(maxbond, ibond, bcoordold, bind, ceng, nsurf, record,
timenew, extf, oldeleno)
! sum up all of the chemical forces

        integer, INTENT(IN):: nsurf, record, ibond, maxbond
        integer,dimension(:,:), INTENT(IN)::bind
        integer, dimension(nsurf,maxbond), INTENT(IN)::oldeleno
        double precision, INTENT(IN)::timenew
        double precision,dimension(nsurf,3), INTENT(IN):: ceng
        double precision,dimension(maxbond,6), INTENT(IN)::bcoordold
        double precision,dimension(nsurf,6), INTENT(INOUT):: extf
        double precision,dimension(3,maxbond)::bforcetrack
        integer :: i, j
        double precision :: z, sigma    !bondlength, spring constant
```

```
double precision, dimension(3)::fbond1,& !bond force on particle 1
          fbond2,&       !bond force on particle/surface 2
          Tbond1,&       !torque due to bond force on particle 1
          Tbond2,&       !torque due to bond force on particle/surface 2
       er           !Lever arm for torque

do i=1,ibond
   z=0.0
   do j=1,3
         z=z+(bcoordold(i,j)-bcoordold(i,j+3))**2
   end do
   z=sqrt(z)
   if (z<lambda) then
                sigma = sigmacompressed
        else
                sigma = sigmatensile
   end if
   do j=1,3
    fbond1(j)=sigma*(z-lambda)*(bcoordold(i,j+3)-bcoordold(i,j))/z
!Eqn. A4 from Hammer and Apte, 1992
    fbond2(j)=-fbond1(j)
        !Torque Arm length
    er(j)=bcoordold(i,j)-ceng(bind(i,1),j)
   end do
        !Calculated by cross product determinant formula
   Tbond1(1)=(er(2)*fbond1(3)-er(3)*fbond1(2))
   Tbond1(2)=(er(3)*fbond1(1)-er(1)*fbond1(3))
   Tbond1(3)=(er(1)*fbond1(2)-er(2)*fbond1(1))
   if (bind(i,2)<=nsurf) then
    do j=1,3
     er(j)=bcoordold(i,j+3)-ceng(bind(i,2),j)
    end do
    Tbond2(1)=(er(2)*fbond2(3)-er(3)*fbond2(2))
    Tbond2(2)=(er(3)*fbond2(1)-er(1)*fbond2(3))
    Tbond2(3)=(er(1)*fbond2(2)-er(2)*fbond2(1))
    do j=1,3
     extf(bind(i,1),j)=extf(bind(i,1),j)+fbond1(j)
     extf(bind(i,2),j)=extf(bind(i,2),j)+fbond2(j)
     extf(bind(i,1),j+3)=extf(bind(i,1),j+3)+Tbond1(j)
     extf(bind(i,2),j+3)=extf(bind(i,2),j+3)+Tbond2(j)
    end do
   else
    do j=1,3
extf(bind(i,1),j)=extf(bind(i,1),j)-fbond1(j)
```

```
                !Torque forces
                extf(bind(i,1),j+3)=extf(bind(i,1),j+3)+Tbond1(j)
              end do
             end if
          !   bforcetrack(1,i)=i ! oldeleno(1,i)
        !  bforcetrack(2,i) = (z-lambda)/abs(z-
lambda)*sqrt(fbond1(1)**2+fbond1(2)**2+fbond1(3)**2)
         ! bforcetrack(3,i) = timenew
          !               write(25,*)  (bforcetrack(j,i),j=1,3)
          !               write(25,*) bforcetrack(2,i), fbond1
          !               write(25,*) fbond1
          !               write(25,*) Tbond1
          !               write(26,*) Tbond1
    !       if (record==1) then
           write(15,*) i, (bcoordold(i,j),j=1,3), (bcoordold(i,j),j=4,6)
            !       write(15,*) z
    !       end if
        end do
!          if (bondbroke) then !   if (bondformed) then   !if (bondformed.or.record==1)
then     !Brownian code
!                do i=1,ibond
!                       write(25,*)  (bforcetrack(j,i),j=1,3)
!                end do
!          end if

END SUBROUTINE Bond_forces

FUNCTION Receptorcoord(iel, vcd, connect)
!This function determines the location of a receptor in an element and provides the
length of that receptor

          double precision, dimension(3) ::  Receptorcoord
          double precision,dimension(:, :), INTENT(IN):: vcd
          integer,dimension(:,:), INTENT(IN) ::connect
          integer :: iel, i, j
          double precision :: signchange1, signchange2  !stores prob values that will
determine sign of eta(1) & (2) for location of receptor
          double precision,dimension(2)::eta !Stores probability values for location of
receptor in element
          double precision,dimension(9,3)::xq !xyz coordinates for all node points for
element iel
          double precision, dimension(3)::x !Stores the coordinates of the location of a
receptor on an element
```

```
        do i = 1,9
                do j = 1,3
                        xq(i,j) = vcd(connect(iel,i),j)
                end do
        end do
        !Randomizing the location of the receptor on element iel
        call Random_Number(eta(1)); call Random_Number(eta(2))
        call Random_Number(signchange1); call Random_Number(signchange2)
        if (signchange1<0.5) eta(1) = -eta(1)
        if (signchange2<0.5) eta(2) = -eta(2)
        call pointx(eta, xq, x)
        Receptorcoord = x

END FUNCTION Receptorcoord

SUBROUTINE pointx(eta, xq, x)
integer::i,j
double precision,dimension(9,3), INTENT(IN)::xq
double precision,dimension(2), INTENT(IN)::eta
double precision::opr,omr,ops,oms,s1,s2,omrr,omss
double precision,dimension(3), INTENT(OUT)::x
double precision,dimension(9)::shape

opr=1.0+eta(1);         omr=1.0-eta(1);         ops=1.0+eta(2);         oms=1.0-eta(2)
do i=1,9
  shape(i)=0.0
end do
shape(1)=omr*oms/4.0
shape(2)=opr*oms/4.0
shape(3)=opr*ops/4.0
shape(4)=omr*ops/4.0
omrr=1.0-eta(1)**2;   omss=1.0-eta(2)**2
shape(5)=omrr*oms/2.0
shape(6)=omss*opr/2.0
shape(7)=omrr*ops/2.0
shape(8)=omss*omr/2.0
shape(1)=shape(1)-shape(5)/2.0-shape(8)/2.0
do i=2,4
  shape(i)=shape(i)-shape(i+3)/2.0-shape(i+4)/2.0
end do
shape(9)=omrr*omss
s1=shape(9)/4.0;         s2=shape(9)/2.0
do i=1,4
  shape(i)=shape(i)+s1
```

```
  shape(i+4)=shape(i+4)-s2
end do
do i=1,3
  x(i)=0.0
end do
do i=1,9
  do j=1,3
    x(j)=x(j)+shape(i)*xq(i,j)
  end do
end do

END SUBROUTINE pointx



END MODULE External_Forces
```

A.4 – MATLAB CODE FOR GENERATION OF SPHERE GEOMETRY

```
function [X,connect]=bemesh(nel)
global ends ang point
% calculates initial positions and connectivity for a QUAD9 spherical
discret.
% nel is number of element-sides per cube-side (Nel=6*nel^2) MUST BE
POWER OF 2
options=optimset('TolFun',1e-10,'TolX',1e-10,'Display','off');

X=zeros(2*(2*nel+1)^2+8*nel*(2*nel-1),3);
connect=int16(zeros(6*nel^2,9));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% nodal coordinates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%corner coords. (x,y,z)
X(1,:)=[-1,-1,-1];
X(1+2*nel,:)=[1,-1,-1];
X((2*nel+1)^2-2*nel,:)=[-1,1,-1];
X((2*nel+1)^2,:)=[1,1,-1];
X((2*nel+1)^2+(2*nel-1)*8*nel+1,:)=[-1,-1,1];
X((2*nel+1)^2+(2*nel-1)*8*nel+1+2*nel,:)=[1,-1,1];
X((2*nel+1)^2+(2*nel-1)*8*nel+(2*nel+1)^2-2*nel,:)=[-1,1,1];
X((2*nel+1)^2+(2*nel-1)*8*nel+(2*nel+1)^2,:)=[1,1,1];
X=X*sqrt(1/3);

%bottom face (z=-
1)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% y=-1 side
X(nel+1,:)=[0,-1,-1]*1/sqrt(2);
for i=2:1+log2(nel)
  for j=1:2^(i-1)
    ends=[X(nel/2^(i-1)+1+(j-1)*nel/2^(i-2)-nel/2^(i-1),:);...
          X(nel/2^(i-1)+1+(j-1)*nel/2^(i-2)+nel/2^(i-1),:)];
    point=2; X(nel/2^(i-1)+1+(j-1)*nel/2^(i-
2),:)=fminsearch('midfuncs',mean(ends),options);
  end
end
% x=-1 side
X(nel*(2*nel+1)+1,:)=[-1,0,-1]*1/sqrt(2);
for i=2:1+log2(nel)
  for j=1:2^(i-1)
    ends=[X((nel/2^(i-1))*(2*nel+1)+1 + ((j-1)*nel/2^(i-
2))*(2*nel+1)...
          -(nel/2^(i-1))*(2*nel+1),:);...
          X((nel/2^(i-1))*(2*nel+1)+1 + ((j-1)*nel/2^(i-
2))*(2*nel+1)...
          +(nel/2^(i-1))*(2*nel+1),:)];
    point=1;
    X((nel/2^(i-1))*(2*nel+1)+1 +...
      ((j-1)*nel/2^(i-
```

```
2))*(2*nel+1),:)=fminsearch('midfuncs',mean(ends),options);
  end
end
% x=+1 side
X((nel+1)*(2*nel+1),:)=[1,0,-1]*1/sqrt(2);
for i=2:1+log2(nel)
  for j=1:2^(i-1)
    ends=[X((nel/2^(i-1)+1)*(2*nel+1) + ((j-1)*nel/2^(i-
2))*(2*nel+1)...
          -(nel/2^(i-1))*(2*nel+1),:);...
          X((nel/2^(i-1)+1)*(2*nel+1) + ((j-1)*nel/2^(i-
2))*(2*nel+1)...
          +(nel/2^(i-1))*(2*nel+1),:)];
    point=1;
    X((nel/2^(i-1)+1)*(2*nel+1) +...
      ((j-1)*nel/2^(i-
2))*(2*nel+1),:)=fminsearch('midfuncs',mean(ends),options);
  end
end
% the other y=constant rows
for k=1:1+2*nel-1
  for i=1:1+log2(nel)
    for j=1:2^(i-1)
      ang=nel/4/nel*abs(k-nel)/2/nel*(abs(j-2^(i-2)-1/2)+1)/2^(i-1);
      if i==1+log2(nel)
        ang=ang/2;
      end
      ends=[X(k*(2*nel+1)+ nel/2^(i-1)+1+(j-1)*nel/2^(i-2)-nel/2^(i-
1),:);...
            X(k*(2*nel+1)+ nel/2^(i-1)+1+(j-1)*nel/2^(i-2)+nel/2^(i-
1),:)];
      X(k*(2*nel+1)+ nel/2^(i-1)+1+(j-1)*nel/2^(i-2),:)...
            =fminsearch('midfunci',mean(ends),options);
    end
  end
end
% y=+1 side
k=2*nel;
X(k*(2*nel+1)+ nel+1,:)=[0,1,-1]*1/sqrt(2);
for i=2:1+log2(nel)
  for j=1:2^(i-1)
    ends=[X(k*(2*nel+1)+ nel/2^(i-1)+1+(j-1)*nel/2^(i-2)-nel/2^(i-
1),:);...
          X(k*(2*nel+1)+ nel/2^(i-1)+1+(j-1)*nel/2^(i-2)+nel/2^(i-
1),:)];
    point=2;
    X(k*(2*nel+1)+ nel/2^(i-1)+1+(j-1)*nel/2^(i-2),:)...
          =fminsearch('midfuncs',mean(ends),options);
  end
end

%y=-1 face
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% rotate bottom nodes about x-axis and then match up new nodes to
first set
Xtemp=X(1:(2*nel+1)^2,:);
[TH,R]=cart2pol(Xtemp(:,2),Xtemp(:,3));
TH=TH-pi/2;
[x,y]=pol2cart(TH,R);
Xtemp(:,2:3)=[x,y];
for i=2:2*nel
  for j=1:2*nel+1
    X((2*nel+1)^2+j+(i-2)*8*nel,:)=Xtemp(j+(2*nel-(i-
1))*(2*nel+1),:);
  end
end
% y=+1 face
% reflect Xtemp across y=0 plane and then match up nodes
Xtemp(:,2)=-Xtemp(:,2);
for i=2:2*nel
  for j=1:2*nel+1
    X((2*nel+1)^2+6*nel-1+j+(i-2)*8*nel,:)=...
                                          Xtemp(j+(2*nel-(i-
1))*(2*nel+1),:);
  end
end
% top face
% reflect bottom face across z=0 and then match up nodes
Xtemp=X(1:(2*nel+1)^2,:);   Xtemp(:,3)=-Xtemp(:,3);
for i=1:2*nel+1
  for j=1:2*nel+1
    X((2*nel+1)^2+8*nel*(2*nel-1)+j+(i-1)*(2*nel+1),:)=...
                                     Xtemp(j+(i-1)*(2*nel+1),:);
  end
end
% side panels (x=constant)
% bottom nodes undergo two rotations so that indices increase as y
inc., z inc.
Xtemp1=X(1:(2*nel+1)^2,:);
[TH,R]=cart2pol(Xtemp1(:,1),Xtemp1(:,3));
TH=TH-pi/2;
[x,y]=pol2cart(TH,R);
Xtemp1=[x,Xtemp1(:,2),y];
[TH,R]=cart2pol(Xtemp1(:,2),Xtemp1(:,3));
TH=TH+pi/2;
[x,y]=pol2cart(TH,R);
Xtemp1=[Xtemp1(:,1),x,y];
Xtemp2=[-Xtemp1(:,1),Xtemp1(:,2:3)];
for i=2:2*nel
  for j=2:2*nel
    X((2*nel+1)^2+(i-1)*(2*nel+1)+(i-2)*(6*nel-1)+2*j-3,:)=...
                                          Xtemp1(j+(i-
1)*(2*nel+1),:);
    X((2*nel+1)^2+(i-1)*(2*nel+1)+(i-2)*(6*nel-1)+2*j-2,:)=...
                                          Xtemp2(j+(i-
1)*(2*nel+1),:);
```

```
  end
end

% connectivity rules
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
% rows in "connect" correspond to elements
% 9 columns of nodal indices: clockwise corners, clockwise sides,
center

% z=-1 face
for i=1:nel
  for j=1:nel
    connect(j+(i-1)*nel,1)=1+2*(j-1)+2*(i-1)*(2*nel+1);
    connect(j+(i-1)*nel,2)=3+2*(j-1)+2*(i-1)*(2*nel+1);
    connect(j+(i-1)*nel,3)=3+2*(j-1)+2*i*(2*nel+1);
    connect(j+(i-1)*nel,4)=1+2*(j-1)+2*i*(2*nel+1);
    connect(j+(i-1)*nel,5)=2+2*(j-1)+2*(i-1)*(2*nel+1);
    connect(j+(i-1)*nel,6)=3+2*(j-1)+(2*(i-1)+1)*(2*nel+1);
    connect(j+(i-1)*nel,7)=2+2*(j-1)+2*i*(2*nel+1);
    connect(j+(i-1)*nel,8)=1+2*(j-1)+(2*(i-1)+1)*(2*nel+1);
    connect(j+(i-1)*nel,9)=2+2*(j-1)+(2*(i-1)+1)*(2*nel+1);
  end
end
% z=+1 face
for i=1:nel
  for j=1:nel
    connect(nel^2+j+(i-1)*nel,1)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +1+2*(j-1)+2*(i-1)*(2*nel+1);
    connect(nel^2+j+(i-1)*nel,2)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +1+2*(j-1)+2*i*(2*nel+1);
    connect(nel^2+j+(i-1)*nel,3)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +3+2*(j-1)+2*i*(2*nel+1);
    connect(nel^2+j+(i-1)*nel,4)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +3+2*(j-1)+2*(i-1)*(2*nel+1);
    connect(nel^2+j+(i-1)*nel,5)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +1+2*(j-1)+(2*(i-1)+1)*(2*nel+1);
    connect(nel^2+j+(i-1)*nel,6)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +2+2*(j-1)+2*i*(2*nel+1);
    connect(nel^2+j+(i-1)*nel,7)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +3+2*(j-1)+(2*(i-1)+1)*(2*nel+1);
    connect(nel^2+j+(i-1)*nel,8)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +2+2*(j-1)+2*(i-1)*(2*nel+1);
    connect(nel^2+j+(i-1)*nel,9)=(2*nel+1)^2+8*nel*(2*nel-1)...
                             +2+2*(j-1)+(2*(i-1)+1)*(2*nel+1);
  end
end
% y=-1 face
for j=1:nel
  connect(2*nel^2+j,1)=1+2*(j-1);
  connect(2*nel^2+j,2)=(2*nel+1)^2 +8*nel+1+2*(j-1);
  connect(2*nel^2+j,3)=(2*nel+1)^2 +8*nel+3+2*(j-1);
```

```
  connect(2*nel^2+j,4)=3+2*(j-1);
  connect(2*nel^2+j,5)=(2*nel+1)^2 +1+2*(j-1);
  connect(2*nel^2+j,6)=(2*nel+1)^2 +8*nel+2+2*(j-1);
  connect(2*nel^2+j,7)=(2*nel+1)^2 +3+2*(j-1);
  connect(2*nel^2+j,8)=2+2*(j-1);
  connect(2*nel^2+j,9)=(2*nel+1)^2 +2+2*(j-1);
end
for i=2:nel
  for j=1:nel
    connect(2*nel^2+j+(i-1)*nel,1)=(2*nel+1)^2 +1+2*(j-1)+(2*i-
3)*8*nel;
    connect(2*nel^2+j+(i-1)*nel,2)=(2*nel+1)^2 +1+2*(j-1)+(2*i-
1)*8*nel;
    connect(2*nel^2+j+(i-1)*nel,3)=(2*nel+1)^2 +3+2*(j-1)+(2*i-
1)*8*nel;
    connect(2*nel^2+j+(i-1)*nel,4)=(2*nel+1)^2 +3+2*(j-1)+(2*i-
3)*8*nel;
    connect(2*nel^2+j+(i-1)*nel,5)=(2*nel+1)^2 +1+2*(j-1)+(2*i-
2)*8*nel;
    connect(2*nel^2+j+(i-1)*nel,6)=(2*nel+1)^2 +2+2*(j-1)+(2*i-
1)*8*nel;
    connect(2*nel^2+j+(i-1)*nel,7)=(2*nel+1)^2 +3+2*(j-1)+(2*i-
2)*8*nel;
    connect(2*nel^2+j+(i-1)*nel,8)=(2*nel+1)^2 +2+2*(j-1)+(2*i-
3)*8*nel;
    connect(2*nel^2+j+(i-1)*nel,9)=(2*nel+1)^2 +2+2*(j-1)+(2*i-
2)*8*nel;
  end
end
% y=+1 face
for i=1:nel-1
  for j=1:nel
    connect(3*nel^2+j+(i-1)*nel,1)=(2*nel+1)*2*nel +1+2*(j-1)+(2*i-
2)*8*nel;
    connect(3*nel^2+j+(i-1)*nel,2)=(2*nel+1)*2*nel +3+2*(j-1)+(2*i-
2)*8*nel;
    connect(3*nel^2+j+(i-1)*nel,3)=(2*nel+1)*2*nel +3+2*(j-
1)+2*i*8*nel;
    connect(3*nel^2+j+(i-1)*nel,4)=(2*nel+1)*2*nel +1+2*(j-
1)+2*i*8*nel;
    connect(3*nel^2+j+(i-1)*nel,5)=(2*nel+1)*2*nel +2+2*(j-1)+(2*i-
2)*8*nel;
    connect(3*nel^2+j+(i-1)*nel,6)=(2*nel+1)*2*nel +3+2*(j-1)+(2*i-
1)*8*nel;
    connect(3*nel^2+j+(i-1)*nel,7)=(2*nel+1)*2*nel +2+2*(j-
1)+2*i*8*nel;
    connect(3*nel^2+j+(i-1)*nel,8)=(2*nel+1)*2*nel +1+2*(j-1)+(2*i-
1)*8*nel;
    connect(3*nel^2+j+(i-1)*nel,9)=(2*nel+1)*2*nel +2+2*(j-1)+(2*i-
1)*8*nel;
  end
end
for j=1:nel
```

```
  connect(3*nel^2+nel*(nel-1)+j,1)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +1+2*(j-1);
  connect(3*nel^2+nel*(nel-1)+j,2)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +3+2*(j-1);
  connect(3*nel^2+nel*(nel-1)+j,3)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +3+2*(j-1)+8*nel+(2*nel+1)^2;
  connect(3*nel^2+nel*(nel-1)+j,4)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +1+2*(j-1)+8*nel+(2*nel+1)^2;
  connect(3*nel^2+nel*(nel-1)+j,5)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +2+2*(j-1);
  connect(3*nel^2+nel*(nel-1)+j,6)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +3+2*(j-1)+8*nel;
  connect(3*nel^2+nel*(nel-1)+j,7)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +2+2*(j-1)+8*nel+(2*nel+1)^2;
  connect(3*nel^2+nel*(nel-1)+j,8)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +1+2*(j-1)+8*nel;
  connect(3*nel^2+nel*(nel-1)+j,9)=(2*nel+1)*2*nel+16*nel*(nel-1)...
                                   +2+2*(j-1)+8*nel;
end
% x=-1 face (exterior elements must be handled separately)
connect(4*nel^2+1,1)=2*(2*nel+1)+1;
connect(4*nel^2+1,2)=(2*nel+1)^2+8*nel+2*nel+4;
connect(4*nel^2+1,3)=(2*nel+1)^2+8*nel+1;
connect(4*nel^2+1,4)=1;
connect(4*nel^2+1,5)=(2*nel+1)^2+2*nel+4;
connect(4*nel^2+1,6)=(2*nel+1)^2+8*nel+2*nel+2;
connect(4*nel^2+1,7)=(2*nel+1)^2+1;
connect(4*nel^2+1,8)=2*nel+2;
connect(4*nel^2+1,9)=(2*nel+1)^2+2*nel+2;
for j=2:nel
  connect(4*nel^2+j,1)=1+2*j*(2*nel+1);
  connect(4*nel^2+j,2)=(2*nel+1)^2+3*(2*nel+1)+2*(2*nel-1)+4*j-1;
  connect(4*nel^2+j,3)=(2*nel+1)^2+3*(2*nel+1)+2*(2*nel-1)+4*j-5;
  connect(4*nel^2+j,4)=1+2*(j-1)*(2*nel+1);
  connect(4*nel^2+j,5)=(2*nel+1)^2+(2*nel+1)+4*j-1;
  connect(4*nel^2+j,6)=(2*nel+1)^2+3*(2*nel+1)+2*(2*nel-1)+4*j-3;
  connect(4*nel^2+j,7)=(2*nel+1)^2+(2*nel+1)+4*j-5;
  connect(4*nel^2+j,8)=1+(2*j-1)*(2*nel+1);
  connect(4*nel^2+j,9)=(2*nel+1)^2+(2*nel+1)+4*j-3;
end
for i=2:nel-1
  connect(4*nel^2+(i-1)*nel+1,1)=(2*nel+1)^2+8*nel*(2*i-
3)+(2*nel+1)+3;
  connect(4*nel^2+(i-1)*nel+1,2)=(2*nel+1)^2+8*nel*(2*i-
1)+(2*nel+1)+3;
  connect(4*nel^2+(i-1)*nel+1,3)=(2*nel+1)^2+8*nel*(2*i-1)+1;
  connect(4*nel^2+(i-1)*nel+1,4)=(2*nel+1)^2+8*nel*(2*i-3)+1;
  connect(4*nel^2+(i-1)*nel+1,5)=(2*nel+1)^2+8*nel*(2*i-
2)+(2*nel+1)+3;
  connect(4*nel^2+(i-1)*nel+1,6)=(2*nel+1)^2+8*nel*(2*i-
1)+(2*nel+1)+1;
  connect(4*nel^2+(i-1)*nel+1,7)=(2*nel+1)^2+8*nel*(2*i-2)+1;
  connect(4*nel^2+(i-1)*nel+1,8)=(2*nel+1)^2+8*nel*(2*i-
```

```
3)+(2*nel+1)+1;
  connect(4*nel^2+(i-1)*nel+1,9)=(2*nel+1)^2+8*nel*(2*i-
2)+(2*nel+1)+1;
end
connect(5*nel^2-(nel-1),1)=(2*nel+1)^2+8*nel*(2*nel-3)+(2*nel+1)+3;
connect(5*nel^2-(nel-1),2)=(2*nel+1)^2+8*nel*(2*nel-1)+2*(2*nel+1)+1;
connect(5*nel^2-(nel-1),3)=(2*nel+1)^2+8*nel*(2*nel-1)+1;
connect(5*nel^2-(nel-1),4)=(2*nel+1)^2+8*nel*(2*nel-3)+1;
connect(5*nel^2-(nel-1),5)=(2*nel+1)^2+8*nel*(2*nel-2)+(2*nel+1)+3;
connect(5*nel^2-(nel-1),6)=(2*nel+1)^2+8*nel*(2*nel-1)+(2*nel+1)+1;
connect(5*nel^2-(nel-1),7)=(2*nel+1)^2+8*nel*(2*nel-2)+1;
connect(5*nel^2-(nel-1),8)=(2*nel+1)^2+8*nel*(2*nel-3)+(2*nel+1)+1;
connect(5*nel^2-(nel-1),9)=(2*nel+1)^2+8*nel*(2*nel-2)+(2*nel+1)+1;
for j=2:nel
  connect(5*nel^2-nel+j,1)=(2*nel+1)^2+8*nel*(2*nel-
3)+(2*nel+1)+4*(j-1)+3;
  connect(5*nel^2-nel+j,2)=(2*nel+1)^2+8*nel*(2*nel-
1)+2*j*(2*nel+1)+1;
  connect(5*nel^2-nel+j,3)=(2*nel+1)^2+8*nel*(2*nel-1)+2*(j-
1)*(2*nel+1)+1;
  connect(5*nel^2-nel+j,4)=(2*nel+1)^2+8*nel*(2*nel-
3)+(2*nel+1)+4*(j-1)-1;
  connect(5*nel^2-nel+j,5)=(2*nel+1)^2+8*nel*(2*nel-
2)+(2*nel+1)+4*(j-1)+3;
  connect(5*nel^2-nel+j,6)=(2*nel+1)^2+8*nel*(2*nel-1)+(2*j-
1)*(2*nel+1)+1;
  connect(5*nel^2-nel+j,7)=(2*nel+1)^2+8*nel*(2*nel-
2)+(2*nel+1)+4*(j-1)-1;
  connect(5*nel^2-nel+j,8)=(2*nel+1)^2+8*nel*(2*nel-
3)+(2*nel+1)+4*(j-1)+1;
  connect(5*nel^2-nel+j,9)=(2*nel+1)^2+8*nel*(2*nel-
2)+(2*nel+1)+4*(j-1)+1;
end
for i=2:nel-1
  for j=2:nel
    connect(4*nel^2+j+(i-1)*nel,1)=(2*nel+1)^2+8*nel*(2*i-
3)+(2*nel+1)+4*j-1;
    connect(4*nel^2+j+(i-1)*nel,2)=(2*nel+1)^2+8*nel*(2*i-
1)+(2*nel+1)+4*j-1;
    connect(4*nel^2+j+(i-1)*nel,3)=...
                                (2*nel+1)^2+8*nel*(2*i-
1)+(2*nel+1)+4*(j-1)-1;
    connect(4*nel^2+j+(i-1)*nel,4)=...
                                (2*nel+1)^2+8*nel*(2*i-
3)+(2*nel+1)+4*(j-1)-1;
    connect(4*nel^2+j+(i-1)*nel,5)=(2*nel+1)^2+8*nel*(2*i-
2)+(2*nel+1)+4*j-1;
    connect(4*nel^2+j+(i-1)*nel,6)=(2*nel+1)^2+8*nel*(2*i-
1)+(2*nel+1)+4*j-3;
    connect(4*nel^2+j+(i-1)*nel,7)=(2*nel+1)^2+8*nel*(2*i-
2)+(2*nel+1)+4*j-5;
    connect(4*nel^2+j+(i-1)*nel,8)=(2*nel+1)^2+8*nel*(2*i-
3)+(2*nel+1)+4*j-3;
```

```
    connect(4*nel^2+j+(i-1)*nel,9)=(2*nel+1)^2+8*nel*(2*i-
2)+(2*nel+1)+4*j-3;
  end
end
% x=+1 face
for j=1:nel-1
  connect(5*nel^2+j,1)=(2*j+1)*(2*nel+1);
  connect(5*nel^2+j,2)=(2*j-1)*(2*nel+1);
  connect(5*nel^2+j,3)=(2*nel+1)^2+8*nel+(2*nel+1)+4*(j-1);
  connect(5*nel^2+j,4)=(2*nel+1)^2+8*nel+(2*nel+1)+4*j;
  connect(5*nel^2+j,5)=2*j*(2*nel+1);
  connect(5*nel^2+j,6)=(2*nel+1)^2+(2*nel+1)+4*(j-1);
  connect(5*nel^2+j,7)=(2*nel+1)^2+8*nel+(2*nel+1)+4*(j-1)+2;
  connect(5*nel^2+j,8)=(2*nel+1)^2+(2*nel+1)+4*j;
  connect(5*nel^2+j,9)=(2*nel+1)^2+(2*nel+1)+4*(j-1)+2;
end
connect(5*nel^2+nel,1)=(2*nel+1)*(2*nel+1);
connect(5*nel^2+nel,2)=(2*nel+1)*(2*nel-1);
connect(5*nel^2+nel,3)=(2*nel+1)^2+8*nel+(2*nel+1)+4*(nel-1);
connect(5*nel^2+nel,4)=(2*nel+1)^2+8*nel*2;
connect(5*nel^2+nel,5)=(2*nel+1)*(2*nel);
connect(5*nel^2+nel,6)=(2*nel+1)^2+(2*nel+1)+4*(nel-1);
connect(5*nel^2+nel,7)=(2*nel+1)^2+8*nel+(2*nel+1)+4*(nel-1)+2;
connect(5*nel^2+nel,8)=(2*nel+1)^2+8*nel;
connect(5*nel^2+nel,9)=(2*nel+1)^2+(2*nel+1)+4*(nel-1)+2;
for i=2:nel-1
  for j=1:nel-1
    connect(5*nel^2+j+(i-1)*nel,1)=(2*nel+1)^2+(2*i-
3)*8*nel+(2*nel+1)+4*j;
    connect(5*nel^2+j+(i-1)*nel,2)=...
                                (2*nel+1)^2+(2*i-
3)*8*nel+(2*nel+1)+4*(j-1);
    connect(5*nel^2+j+(i-1)*nel,3)=...
                                (2*nel+1)^2+(2*i-
1)*8*nel+(2*nel+1)+4*(j-1);
    connect(5*nel^2+j+(i-1)*nel,4)=...
                                (2*nel+1)^2+(2*i-
1)*8*nel+(2*nel+1)+4*(j-1)+4;
    connect(5*nel^2+j+(i-1)*nel,5)=(2*nel+1)^2+(2*i-
3)*8*nel+(2*nel+1)+4*j-2;
    connect(5*nel^2+j+(i-1)*nel,6)=...
                                (2*nel+1)^2+(2*i-
2)*8*nel+(2*nel+1)+4*(j-1);
    connect(5*nel^2+j+(i-1)*nel,7)=...
                                (2*nel+1)^2+(2*i-
1)*8*nel+(2*nel+1)+4*(j-1)+2;
    connect(5*nel^2+j+(i-1)*nel,8)=(2*nel+1)^2+(2*i-
2)*8*nel+(2*nel+1)+4*j;
    connect(5*nel^2+j+(i-1)*nel,9)=(2*nel+1)^2+(2*i-
2)*8*nel+(2*nel+1)+4*j-2;
  end
end
for i=2:nel-1
```

```
  connect(5*nel^2+nel+(i-1)*nel,1)=(2*nel+1)^2+2*(i-1)*8*nel;
  connect(5*nel^2+nel+(i-1)*nel,2)=...
                                (2*nel+1)^2+(2*i-
3)*8*nel+(2*nel+1)+4*(nel-1);
  connect(5*nel^2+nel+(i-1)*nel,3)=...
                                (2*nel+1)^2+(2*i-
1)*8*nel+(2*nel+1)+4*(nel-1);
  connect(5*nel^2+nel+(i-1)*nel,4)=(2*nel+1)^2+2*i*8*nel;
  connect(5*nel^2+nel+(i-1)*nel,5)=...
                                (2*nel+1)^2+(2*i-
3)*8*nel+(2*nel+1)+4*nel-2;
  connect(5*nel^2+nel+(i-1)*nel,6)=...
                                (2*nel+1)^2+(2*i-
2)*8*nel+(2*nel+1)+4*(nel-1);
  connect(5*nel^2+nel+(i-1)*nel,7)=...
                                (2*nel+1)^2+(2*i-
1)*8*nel+(2*nel+1)+4*(nel-1)+2;
  connect(5*nel^2+nel+(i-1)*nel,8)=(2*nel+1)^2+(2*i-1)*8*nel;
  connect(5*nel^2+nel+(i-1)*nel,9)=...
                                (2*nel+1)^2+(2*i-
2)*8*nel+(2*nel+1)+4*(nel-1);
end
for j=1:nel-1
  connect(5*nel^2+j+(nel-1)*nel,1)=(2*nel+1)^2+(2*nel-
3)*8*nel+(2*nel+1)+4*j;
  connect(5*nel^2+j+(nel-1)*nel,2)=...
                                 (2*nel+1)^2+(2*nel-
3)*8*nel+(2*nel+1)+4*(j-1);
  connect(5*nel^2+j+(nel-1)*nel,3)=...
                                 (2*nel+1)^2+(2*nel-
1)*8*nel+(2*nel+1)*(2*j-1);
  connect(5*nel^2+j+(nel-1)*nel,4)=...
                                 (2*nel+1)^2+(2*nel-
1)*8*nel+(2*j+1)*(2*nel+1);
  connect(5*nel^2+j+(nel-1)*nel,5)=(2*nel+1)^2+(2*nel-
3)*8*nel+(2*nel+1)+4*j-2;
  connect(5*nel^2+j+(nel-1)*nel,6)=...
                                 (2*nel+1)^2+(2*nel-
2)*8*nel+(2*nel+1)+4*(j-1);
  connect(5*nel^2+j+(nel-1)*nel,7)=...
                                 (2*nel+1)^2+(2*nel-
1)*8*nel+(2*nel+1)*2*j;
  connect(5*nel^2+j+(nel-1)*nel,8)=(2*nel+1)^2+(2*nel-
2)*8*nel+(2*nel+1)+4*j;
  connect(5*nel^2+j+(nel-1)*nel,9)=...
                                 (2*nel+1)^2+(2*nel-
2)*8*nel+(2*nel+1)+4*(j-1)+2;
end
connect(6*nel^2,1)=(2*nel+1)^2+(2*nel-2)*8*nel;
connect(6*nel^2,2)=(2*nel+1)^2+(2*nel-3)*8*nel+(2*nel+1)+4*(nel-1);
connect(6*nel^2,3)=(2*nel+1)^2+(2*nel-1)*8*nel+(2*nel+1)*(2*nel-1);
connect(6*nel^2,4)=(2*nel+1)^2+(2*nel-1)*8*nel+(2*nel+1)*(2*nel+1);
connect(6*nel^2,5)=(2*nel+1)^2+(2*nel-3)*8*nel+(2*nel+1)+4*nel-2;
```

```
connect(6*nel^2,6)=(2*nel+1)^2+(2*nel-2)*8*nel+(2*nel+1)+4*(nel-1);
connect(6*nel^2,7)=(2*nel+1)^2+(2*nel-1)*8*nel+(2*nel+1)*2*nel;
connect(6*nel^2,8)=(2*nel+1)^2+(2*nel-1)*8*nel;
connect(6*nel^2,9)=(2*nel+1)^2+(2*nel-2)*8*nel+(2*nel+1)+4*(nel-1)+2;
```

A.5 – MATLAB CODE FOR GENERATION OF PARTICLE AGGREGATE GEOMETRIES

```
function [CTMconnect, CTMmesh, newRadius, particle2] =
meshAdjust4(nel)
%Reads in coordinates created by bemesh and shifts vector magnitudes
to
%create new geometry
%This creates a 4-mer in a diamond formation using commands 1 -> 5 ->
0
%This creates a 4-mer in a linear chain using commands 5 -> 6 -> 0

%Generate spherical mesh
[meshCoords, connect] = bemesh(nel);
connect = double(connect);
%corrections for an error in the connect file node assignment
connect(88,9) = 168;
connect(92,9) = 232;

%Set the scaling for the nodes at the plane of intersection in the
minor
%axes - affects the look of the "pinch" between two monomers
pf = 1.25; %cross-setional area of plane of intersection
%1.25 = 125% of original area

%Determine the coordinates of corner points along x, y, and z axis
[rowmax,colmax] = find(meshCoords > 0.99);
[rowmin,colmin] = find(meshCoords < -0.99);
maxheight = meshCoords(rowmax,colmax);
minheight = meshCoords(rowmin,colmin);

%Determine the center and radius of the sphere
distances = [abs(diag(minheight)); abs(diag(maxheight))];
radius = mean(distances);
% center = zeros(1,3);
% center(1,1) = (maxheight(3,1) + minheight(3,1))/2;
% center(1,2) = (maxheight(3,2) + minheight(3,2))/2;
% center(1,3) = (maxheight(3,3) + minheight(3,3))/2;

%Prompt user to remove sides from particle for meshing
[m,n] = size(meshCoords);
facesRemoved = 0;
newMesh = meshCoords;
response = 1;
i = 1;
while response ~=0
    response = input('Select which face to remove from particle. Type
''0'' to end.\n');
    switch response
        case 1 %new monomer attached to bottom face
```

```
            for i1 = 1:m
                if meshCoords(i1,3) < -0.8586 %on bottom face
                    if meshCoords(i1,1) > -0.3659 && meshCoords(i1,1)
< 0.3659
                        if meshCoords(i1,2) > -0.3591 &&
meshCoords(i1,2) < 0.3591
                            newMesh(i1,:) = NaN;
                        end
                    end
                end
            end
            emptyFaces(i) = 1;
        case 2 %new monomer attached to back face
            for i2 = 1:m
                if meshCoords(i2,1) < -0.8586 %on back face
                    if meshCoords(i2,2) > -0.3659 && meshCoords(i2,2)
< 0.3659
                        if meshCoords(i2,3) > -0.3591 &&
meshCoords(i2,3) < 0.3591
                            newMesh(i2,:) = NaN;
                        end
                    end
                end
            end
            emptyFaces(i) = 2;
        case 3 %new monomer attached to top face
            for i3 = 1:m
                if meshCoords(i3,3) > 0.8586 %on top face
                    if meshCoords(i3,1) > -0.3659 && meshCoords(i3,1)
< 0.3659
                        if meshCoords(i3,2) > -0.3591 &&
meshCoords(i3,2) < 0.3591
                            newMesh(i3,:) = NaN;
                        end
                    end
                end
            end
            emptyFaces(i) = 3;
        case 4 %new monomer attached to front face
            for i4 = 1:m
                if meshCoords(i4,1) > 0.8586 %on front face
                    if meshCoords(i4,2) > -0.3659 && meshCoords(i4,2)
< 0.3659
                        if meshCoords(i4,3) > -0.3591 &&
meshCoords(i4,3) < 0.3591
                            newMesh(i4,:) = NaN;
                        end
                    end
                end
            end
            emptyFaces(i) = 4;
        case 5 %new monommer attached to left face
            for i5 = 1:m
```

```
                if meshCoords(i5,2) > 0.8586 %on left face
                    if meshCoords(i5,1) > -0.3659 && meshCoords(i5,1)
< 0.3659
                        if meshCoords(i5,3) > -0.3591 &&
meshCoords(i5,3) < 0.3591
                            newMesh(i5,:) = NaN;
                        end
                    end
                end
            end
            emptyFaces(i) = 5;
        case 6 %new monomer attached to right face
            for i6 = 1:m
                if meshCoords(i6,2) < -0.8586 %on right face
                    if meshCoords(i6,1) > -0.3659 && meshCoords(i6,1)
< 0.3659
                        if meshCoords(i6,3) > -0.3591 &&
meshCoords(i6,3) < 0.3591
                            newMesh(i6,:) = NaN;
                        end
                    end
                end
            end
            emptyFaces(i) = 6;
        otherwise
            break
    end
    facesRemoved = facesRemoved + 1;
    i = i+1;
end

%plot particle with removed faces
%
connect2=[connect(:,1),connect(:,5),connect(:,2),connect(:,6),connect
(:,3),...
%           connect(:,7),connect(:,4),connect(:,8)];
% figure;
% particle = patch('faces',connect2,'vertices',newMesh,'FaceColor',[1
.5 .5],'EdgeColor',[1 0 0]);

%%%Check number of faces removed and attach particle copies to
them%%%
if facesRemoved > 0
    CTMmesh = zeros(m*(facesRemoved+1),n);
    for f = 1:size(emptyFaces,2)
        new2Mesh(:,:,f) = meshCoords;
        switch emptyFaces(f)
            case 1 %open face is on bottom
                for i3 = 1:m %loop for particle copy to have open top
face
                    if meshCoords(i3,3) > 0.8586
                        if meshCoords(i3,1) > -0.3591 &&
meshCoords(i3,1) < 0.3591
```

```
                            if meshCoords(i3,2) > -0.3659 &&
meshCoords(i3,2) < 0.3659
                                new2Mesh(i3,:,f) = NaN;
                            end
                        end
                    end
                end

                %for 4-mer15, face 5 needs to be removed as well
                if isequal(emptyFaces,[1,5]) ||
isequal(emptyFaces,[5,1])
                    for i5 = 1:m
                        if meshCoords(i5,2) > 0.8586 %on left face
                            if meshCoords(i5,1) > -0.3659 &&
meshCoords(i5,1) < 0.3659
                                if meshCoords(i5,3) > -0.3591 &&
meshCoords(i5,3) < 0.3591
                                    new2Mesh(i5,:,f) = NaN;
                                end
                            end
                        end
                    end
                end

                %combine coordinates
                new2Mesh(:,3,f) = new2Mesh(:,3,f) - (radius +
0.8586);
                if f == 1
                    addition = [newMesh; new2Mesh(:,:,f)];
                    [addRow,~] = size(addition);
                    CTMmesh(1:addRow,:) = [newMesh; new2Mesh(:,:,f)];
                else
                    CTMmesh(1+(f*m):((f+1)*m),:) = new2Mesh(:,:,f);
                end

                %make new connect array by stacking two and
restarting count for second set
                if f == 1
                    CTMconnect = [connect; connect+size(newMesh,1)];
                else
                    CTMconnect = [CTMconnect;
connect+size(newMesh,1)*f];
                end

                %shift z-coordinate of vertices at intersecting face
so
                %they connect
                for i = 1:size(CTMmesh,1)
                    if CTMmesh(i,3) <= -0.8500 && CTMmesh(i,3) >= -
1.0100 %points lying on edge of removed face in z-direction
```

```
                        if CTMmesh(i,2) <= 0.3700 && CTMmesh(i,2) >=
-0.3700 && CTMmesh(i,1) >= -0.3833 && CTMmesh(i,1) <= 0.3833 %only
points at desired particle interface
                            CTMmesh(i,1) = CTMmesh(i,1)*pf; %can toy
with if i want to adjust pinch
                            CTMmesh(i,2) = CTMmesh(i,2)*pf; %can toy
with if i want to adjust pinch
                            CTMmesh(i,3) = -0.9293;
                        end
                    end
                end

                %remove duplicate coordinates on plane of
intersection
                [row,~] = find(CTMmesh(:,3) == -0.9293);
                [correctRow,~] = find(CTMmesh(row,2) <= 0.3700*pf &
CTMmesh(row,2) >= -0.3700*pf & CTMmesh(row,1) <= 0.3833*pf &
CTMmesh(row,1) >= -0.3833*pf); %only look on desired particle
interface
                intersect = CTMmesh(row(correctRow),:);
                for i = 1:size(intersect,1)
                    if isnan(intersect(i,:))
                        continue
                    end
                    tf = ismember(intersect,intersect(i,:),'rows');
                    ind = find(tf == 1);
                    if numel(ind) > 1
                        CTMmesh(row(ind(2)),:) = NaN;
                        intersect(ind(2),:) = NaN; %just for
diagnostics
                        %loop over connect array and renumber
vertices
                        for x = 1:size(CTMconnect,1)
                            for y = 1:size(CTMconnect,2)
                                if CTMconnect(x,y) == row(ind(2))
                                    CTMconnect(x,y) = row(ind(1));
                                end
                            end
                        end
                    end
                end

            case 2 %open face is on back
                for i4 = 1:m %loop for particle copy to have open
front face
                    if meshCoords(i4,1) > 0.8586
                        if meshCoords(i4,2) > -0.3659 &&
meshCoords(i4,2) < 0.3659
                            if meshCoords(i4,3) > -0.3591 &&
meshCoords(i4,3) < 0.3591
                                new2Mesh(i4,:,f) = NaN;
                            end
                        end
```

```
                end
            end

            %combine coordinates
            new2Mesh(:,1,f) = new2Mesh(:,1,f) - (radius +
0.8586);
            if f == 1
                addition = [newMesh; new2Mesh(:,:,f)];
                [addRow,~] = size(addition);
                CTMmesh(1:addRow,:) = [newMesh; new2Mesh(:,:,f)];
            else
                CTMmesh(1+(f*m):((f+1)*m),:) = new2Mesh(:,:,f);
            end

            %make new connect array by stacking two and
restarting count for second set
            if f == 1
                CTMconnect = [connect; connect+size(newMesh,1)];
            else
                CTMconnect = [CTMconnect;
connect+size(newMesh,1)*f];
            end

            %shift x-coordinate of vertices at intersecting face
so
            %they connect
            for i = 1:size(CTMmesh,1)
                if CTMmesh(i,1) >= -1.0100 && CTMmesh(i,1) <= -
0.8500 %points lying on edge of removed vace in z-direction
                    if CTMmesh(i,3) <= 0.3700 && CTMmesh(i,3) >=
-0.3700 && CTMmesh(i,2) >= -0.3833 && CTMmesh(i,2) <= 0.3833 %only
points at desired particle interface
                        CTMmesh(i,1) = -0.9293;
                        CTMmesh(i,2) = CTMmesh(i,2)*pf; %can toy
with if i want to adjust pinch
                        CTMmesh(i,3) = CTMmesh(i,3)*pf; %can toy
with if i want to adjust pinch
                    end
                end
            end

            %remove duplicate coordinates on plane of
intersection
            [row,~] = find(CTMmesh(:,1) == -0.9293);
            [correctRow,~] = find(CTMmesh(row,3) <= 0.3700*pf &
CTMmesh(row,3) >= -0.3700*pf & CTMmesh(row,2) >= -0.3833*pf &
CTMmesh(row,2) <= 0.3833*pf); %only look on desired particle
interface
            intersect = CTMmesh(row(correctRow),:);
            for i = 1:size(intersect,1)
                if isnan(intersect(i,:))
                    continue
```

```
                    end
                    tf = ismember(intersect,intersect(i,:),'rows');
                    ind = find(tf == 1);
                    if numel(ind) > 1
                        CTMmesh(row(ind(2)),:) = NaN;
                        intersect(ind(2),:) = NaN; %just for
diagnostics
                        %loop over connect array and renumber
vertices
                        for x = 1:size(CTMconnect,1)
                            for y = 1:size(CTMconnect,2)
                                if CTMconnect(x,y) == row(ind(2))
                                    CTMconnect(x,y) = row(ind(1));
                                end
                            end
                        end
                    end
                end


            case 3 %open face is on top
                for i1 = 1:m %loop for particle copy to have open
bottom face
                    if meshCoords(i1,3) < -0.8586
                        if meshCoords(i1,1) > -0.3659 &&
meshCoords(i1,1) < 0.3659
                            if meshCoords(i1,2) > -0.3591 &&
meshCoords(i1,2) < 0.3591
                                new2Mesh(i1,:,f) = NaN;
                            end
                        end
                    end
                end

                %combine coordinates
                new2Mesh(:,3,f) = new2Mesh(:,3,f) + (radius +
0.8586);
                if f == 1
                    addition = [newMesh; new2Mesh(:,:,f)];
                    [addRow,~] = size(addition);
                    CTMmesh(1:addRow,:) = [newMesh; new2Mesh(:,:,f)];
                else
                    CTMmesh(1+(f*m):((f+1)*m),:) = new2Mesh(:,:,f);
                end

                %make new connect array by stacking two and
restarting count for second set
                if f == 1
                    CTMconnect = [connect; connect+size(newMesh,1)];
                else
                    CTMconnect = [CTMconnect;
connect+size(newMesh,1)*f];
```

```
                end

                %shift z-coordinate of vertices at intersecting face
so
                %they connect
                for i = 1:size(CTMmesh,1)
                    if CTMmesh(i,3) >= 0.8500 && CTMmesh(i,3) <= 1.01
%points lying on edge of removed vace in z-direction
                        if CTMmesh(i,2) <= 0.3700 && CTMmesh(i,2) >=
-0.3700 && CTMmesh(i,1) >= -0.3833 && CTMmesh(i,1) <= 0.3833 %only
points at desired particle interface
                            CTMmesh(i,1) = CTMmesh(i,1)*pf; %can toy
with if i want to adjust pinch
                            CTMmesh(i,2) = CTMmesh(i,2)*pf; %can toy
with if i want to adjust pinch
                            CTMmesh(i,3) = 0.9293;
                        end
                    end
                end

                %remove duplicate coordinates on plane of
intersection
                [row,~] = find(CTMmesh(:,3) == 0.9293);
                [correctRow,~] = find(CTMmesh(row,2) <= 0.3700*pf &
CTMmesh(row,2) >= -0.3700*pf & CTMmesh(row,1) >= -0.3833*pf &
CTMmesh(row,1) <= 0.3833*pf); %only look on desired particle
interface
                intersect = CTMmesh(row(correctRow),:);
                for i = 1:size(intersect,1)
                    if isnan(intersect(i,:))
                        continue
                    end
                    tf = ismember(intersect,intersect(i,:),'rows');
                    ind = find(tf == 1);
                    if numel(ind) > 1
                        CTMmesh(row(ind(2)),:) = NaN;
                        intersect(ind(2),:) = NaN; %just for
diagnostics
                        %loop over connect array and renumber
vertices
                        for x = 1:size(CTMconnect,1)
                            for y = 1:size(CTMconnect,2)
                                if CTMconnect(x,y) == row(ind(2))
                                    CTMconnect(x,y) = row(ind(1));
                                end
                            end
                        end
                    end
                end

            case 4 %open face is on front
                for i2 = 1:m %loop for particle copy to have open
back face
```

```
                    if meshCoords(i2,1) < -0.8586
                        if meshCoords(i2,2) > -0.3659 &&
meshCoords(i2,2) < 0.3659
                            if meshCoords(i2,3) > -0.3591 &&
meshCoords(i2,3) < 0.3591
                                new2Mesh(i2,:,f) = NaN;
                            end
                        end
                    end
                end

                %for 4-mer24, face 4 needs to be removed as well
                if isequal(emptyFaces,[2,4]) ||
isequal(emptyFaces,[4,2])
                    for i4 = 1:m
                        if meshCoords(i4,1) > 0.8586 %on front face
                            if meshCoords(i4,2) > -0.3659 &&
meshCoords(i4,2) < 0.3659
                                if meshCoords(i4,3) > -0.3591 &&
meshCoords(i4,3) < 0.3591
                                    new2Mesh(i4,:,f) = NaN;
                                end
                            end
                        end
                    end
                end

                %combine coordinates
                new2Mesh(:,1,f) = new2Mesh(:,1,f) + (radius +
0.8586);
                if f == 1
                    addition = [newMesh; new2Mesh(:,:,f)];
                    [addRow,~] = size(addition);
                    CTMmesh(1:addRow,:) = [newMesh; new2Mesh(:,:,f)];
                else
                    CTMmesh(1+(f*m):((f+1)*m),:) = new2Mesh(:,:,f);
                end

                %make new connect array by stacking two and
restarting count for second set
                if f == 1
                    CTMconnect = [connect; connect+size(newMesh,1)];
                else
                    CTMconnect = [CTMconnect;
connect+size(newMesh,1)*f];
                end

                %shift x-coordinate of vertices at intersecting face
so
                %they connect
                for i = 1:size(CTMmesh,1)
```

```
                    if CTMmesh(i,1) >= 0.8500 && CTMmesh(i,1) <=
1.0100 %points lying on edge of removed vace in z-direction
                        if CTMmesh(i,3) <= 0.3700 && CTMmesh(i,3) >=
-0.3700 && CTMmesh(i,2) >= -0.3833 && CTMmesh(i,2) <= 0.3833 %only
points at desired particle interface
                            CTMmesh(i,1) = 0.9293;
                            CTMmesh(i,2) = CTMmesh(i,2)*pf; %can toy
with if i want to adjust pinch
                            CTMmesh(i,3) = CTMmesh(i,3)*pf; %can toy
with if i want to adjust pinch
                        end
                    end
                end

                %remove duplicate coordinates on plane of
intersection
                [row,~] = find(CTMmesh(:,1) == 0.9293);
                [correctRow,~] = find(CTMmesh(row,3) <= 0.3700*pf &
CTMmesh(row,3) >= -0.3700*pf & CTMmesh(row,2) >= -0.3833*pf &
CTMmesh(row,2) <= 0.3833*pf); %only look on desired particle
interface
                intersect = CTMmesh(row(correctRow),:);
                for i = 1:size(intersect,1)
                    if isnan(intersect(i,:))
                        continue
                    end
                    tf = ismember(intersect,intersect(i,:),'rows');
                    ind = find(tf == 1);
                    if numel(ind) > 1
                        CTMmesh(row(ind(2)),:) = NaN;
                        intersect(ind(2),:) = NaN; %just for
diagnostics
                        %loop over connect array and renumber
vertices
                        for x = 1:size(CTMconnect,1)
                            for y = 1:size(CTMconnect,2)
                                if CTMconnect(x,y) == row(ind(2))
                                    CTMconnect(x,y) = row(ind(1));
                                end
                            end
                        end
                    end
                end

            case 5 %missing face is on left
                for i6 = 1:m %loop for particle copy to have open
right face
                    if meshCoords(i6,2) < -0.8586
                        if meshCoords(i6,1) > -0.3659 &&
meshCoords(i6,1) < 0.3659
                            if meshCoords(i6,3) > -0.3591 &&
meshCoords(i6,3) < 0.3591
                                new2Mesh(i6,:,f) = NaN;
```

```
                        end
                    end
                end
            end

            %for 4-mer15, face 1 needs to be removed as well
            if isequal(emptyFaces,[1,5]) ||
isequal(emptyFaces,[5,1])
                for i1 = 1:m
                    if meshCoords(i1,3) < -0.8586 %on bottom face
                        if meshCoords(i1,1) > -0.3659 &&
meshCoords(i1,1) < 0.3659
                            if meshCoords(i1,2) > -0.3591 &&
meshCoords(i1,2) < 0.3591
                                new2Mesh(i1,:,f) = NaN;
                            end
                        end
                    end
                end
            end

            %for 4-mer56, face 5 needs to be removed as well
            if isequal(emptyFaces,[5,6]) ||
isequal(emptyFaces,[6,5])
                for i5 = 1:m
                    if meshCoords(i5,2) > 0.8586 %on left face
                        if meshCoords(i5,1) > -0.3659 &&
meshCoords(i5,1) < 0.3659
                            if meshCoords(i5,3) > -0.3591 &&
meshCoords(i5,3) < 0.3591
                                new2Mesh(i5,:,f) = NaN;
                            end
                        end
                    end
                end
            end

            %combine coordinates
            new2Mesh(:,2,f) = new2Mesh(:,2,f) + (radius +
0.8586);
            if f == 1
                addition = [newMesh; new2Mesh(:,:,f)];
                [addRow,~] = size(addition);
                CTMmesh(1:addRow,:) = [newMesh; new2Mesh(:,:,f)];
            else
                CTMmesh(1+(f*m):((f+1)*m),:) = new2Mesh(:,:,f);
            end

            %make new connect array by stacking two and
restarting count for second set
            if f == 1
                CTMconnect = [connect; connect+size(newMesh,1)];
```

```
                else
                    CTMconnect = [CTMconnect;
connect+size(newMesh,1)*f];
                end

                %shift y-coordinate of vertices at intersecting face
so
                %they connect
                for i = 1:size(CTMmesh,1)
                    if CTMmesh(i,2) >= 0.8500 && CTMmesh(i,2) <=
1.0100
                        if CTMmesh(i,3) >= -0.3700 && CTMmesh(i,3) <=
0.3700 && CTMmesh(i,1) >= -0.3833 && CTMmesh(i,1) <= 0.3833 %only
points at desired particle interface
                            CTMmesh(i,1) = CTMmesh(i,1)*pf; %can toy
with if i want to adjust pinch
                            CTMmesh(i,2) = 0.9293;
                            CTMmesh(i,3) = CTMmesh(i,3)*pf; %can toy
with if i want to adjust pinch
                        end
                    end
                end

                %remove duplicate coordinates on plane of
intersection
                [row,~] = find(CTMmesh(:,2) == 0.9293); %only for
desired particle
                [correctRow,~] = find(CTMmesh(row,1) <= 0.3833*pf &
CTMmesh(row,1) >= -0.3833*pf & CTMmesh(row,3) >= -0.3700*pf &
CTMmesh(row,3) <= 0.3700*pf); %only look on desired particle
interface
                intersect = CTMmesh(row(correctRow),:);
                for i = 1:size(intersect,1)
                    if isnan(intersect(i,:))
                        continue
                    end
                    tf = ismember(intersect,intersect(i,:),'rows');
                    ind = find(tf == 1);
                    if numel(ind) > 1
                        CTMmesh(row(ind(2)),:) = NaN;
                        intersect(ind(2),:) = NaN; %just for
diagnostics
                        %loop over connect array and renumber
vertices
                        for x = 1:size(CTMconnect,1)
                            for y = 1:size(CTMconnect,2)
                                if CTMconnect(x,y) == row(ind(2))
                                    CTMconnect(x,y) = row(ind(1));
                                end
                            end
                        end
                    end
                end
```

```
            case 6 %missing face is on right
                for i5 = 1:m %loop for particle copy to have open
left face
                    if meshCoords(i5,2) > 0.8586
                        if meshCoords(i5,1) > -0.3659 &&
meshCoords(i5,1) < 0.3659
                            if meshCoords(i5,3) > -0.3591 &&
meshCoords(i5,3) < 0.3591
                                new2Mesh(i5,:,f) = NaN;
                            end
                        end
                    end
                end

                %combine coordinates
                new2Mesh(:,2,f) = new2Mesh(:,2,f) - (radius +
0.8586);
                if f == 1
                    addition = [newMesh; new2Mesh(:,:,f)];
                    [addRow,~] = size(addition);
                    CTMmesh(1:addRow,:) = [newMesh; new2Mesh(:,:,f)];
                else
                    CTMmesh(1+(f*m):((f+1)*m),:) = new2Mesh(:,:,f);
                end

                %make new connect array by stacking two and
restarting count for second set
                if f == 1
                    CTMconnect = [connect; connect+size(newMesh,1)];
                else
                    CTMconnect = [CTMconnect;
connect+size(newMesh,1)*f];
                end

                %shift y-coordinate of vertices at intersecting face
so
                %they connect
                for i = 1:size(CTMmesh,1)
                    if CTMmesh(i,2) <= -0.8500 && CTMmesh(i,2) >= -
1.0100
                        if CTMmesh(i,3) <= 0.3700 && CTMmesh(i,3) >=
-0.3700 && CTMmesh(i,1) <= 0.3833 && CTMmesh(i,1) >= -0.3833 %only
points at desired particle interface
                            CTMmesh(i,1) = CTMmesh(i,1)*pf; %can toy
with if i want to adjust pinch
                            CTMmesh(i,2) = -0.9293;
                            CTMmesh(i,3) = CTMmesh(i,3)*pf; %can toy
with if i want to adjust pinch
                        end
                    end
                end
```

```
                %remove duplicate coordinates on plane of
intersection
                [row,~] = find(CTMmesh(:,2) == -0.9293); %only for
desired particle
                [correctRow,~] = find(CTMmesh(row,1) <= 0.3833*pf &
CTMmesh(row,1) >= -0.3833*pf & CTMmesh(row,3) >= -0.3700*pf &
CTMmesh(row,3) <= 0.3700*pf); %only look on desired particle
interface
                intersect = CTMmesh(row(correctRow),:);
                for i = 1:size(intersect,1)
                    if isnan(intersect(i,:))
                        continue
                    end
                    tf = ismember(intersect,intersect(i,:),'rows');
                    ind = find(tf == 1);
                    if numel(ind) > 1
                        CTMmesh(row(ind(2)),:) = NaN;
                        intersect(ind(2),:) = NaN; %just for
diagnostics
                        %loop over connect array and renumber
vertices
                        for x = 1:size(CTMconnect,1)
                            for y = 1:size(CTMconnect,2)
                                if CTMconnect(x,y) == row(ind(2))
                                    CTMconnect(x,y) = row(ind(1));
                                end
                            end
                        end
                    end
                end
        end
    end

%%%If faces 1 and 5 were removed, adds monomer to complete 4-mer%%%
    if isequal(emptyFaces,[1,5]) || isequal(emptyFaces,[5,1])
        %fourth monomer should have faces 3 and 6 removed
        new3Mesh = meshCoords;
        %removing face 3
        for i3 = 1:m
            if meshCoords(i3,3) > 0.8586 %on top face
                if meshCoords(i3,1) > -0.3659 && meshCoords(i3,1) <
0.3659
                    if meshCoords(i3,2) > -0.3591 && meshCoords(i3,2)
< 0.3591
                        new3Mesh(i3,:) = NaN;
                    end
                end
            end
        end
        %removing face 6
        for i6 = 1:m
            if meshCoords(i6,2) < -0.8586 %on right face
```

```
                if meshCoords(i6,1) > -0.3659 && meshCoords(i6,1) <
0.3659
                    if meshCoords(i6,3) > -0.3591 && meshCoords(i6,3)
< 0.3591
                        new3Mesh(i6,:) = NaN;
                    end
                end
            end
        end

        %combine coordinates
        new3Mesh(:,2) = new3Mesh(:,2) + (radius + 0.8586);
        new3Mesh(:,3) = new3Mesh(:,3) - (radius + 0.8586);
        CTMmesh = [CTMmesh ; zeros(m,n)];
        CTMmesh(1+((f+1)*m):((f+2)*m),:) = new3Mesh;
        CTMconnect = [CTMconnect; connect+size(newMesh,1)*(f+1)];

        %if adjusting the pinch on the 4-mer, cannot just scale
nodes*pf,
        %since the 4th monomer center is no longer zero. Must
increase
        %nodes above the monomers center by 25%, and decrease nodes
below
        %center by 25%
        moncenter = [0 0.9293*2 -0.9293*2];
        scale = 0.25; %difference between pf and 1

        %shift z-coordinate of vertices at intersecting face so they
connect
        for i = 1:size(CTMmesh,1)
            if CTMmesh(i,3) <= -0.8500 && CTMmesh(i,3) >= -1.01
%intersecting plane for 4th monomer along z-axis
                if CTMmesh(i,2) <= 2.2286 && CTMmesh(i,2) >= 1.4886
%bounds of case 1 for initial loop + (radius + 0.8586)
                    CTMmesh(i,1) = CTMmesh(i,1)*pf; %can toy with if
i want to adjust pinch
                    if CTMmesh(i,2) > moncenter(2)
                        CTMmesh(i,2) = CTMmesh(i,2)+((CTMmesh(i,2)-
(radius + 0.8586))*scale); %can toy with if i want to adjust pinch
                    end
                    if CTMmesh(i,2) < moncenter(2)
                        CTMmesh(i,2) = CTMmesh(i,2)-
(abs(CTMmesh(i,2)-(radius + 0.8586))*scale); %can toy with if i want
to adjust pinch
                    end
                    CTMmesh(i,3) = -0.9293;
                end
            end
        end

        %remove duplicate coordinates on z-plane of intersection
```

```
        [row,~] = find(CTMmesh(:,3) == -0.9293 & CTMmesh(:,2) >=
1.3900); %look only on desired plane of intersection
        [correctRow,~] = find(CTMmesh(row,2) <= 2.3300 &
CTMmesh(row,2) >= 1.3900); %only look on desired particle interface
        intersect = CTMmesh(row(correctRow),:);
        for i = 1:size(intersect,1)
            if isnan(intersect(i,:))
                continue
            end
            tf = ismember(intersect,intersect(i,:),'rows');
            ind = find(tf == 1);
            if numel(ind) > 1
                CTMmesh(row(ind(2)),:) = NaN;
                intersect(ind(2),:) = NaN; %just for diagnostics
                %loop over connect array and renumber vertices
                for x = 1:size(CTMconnect,1)
                    for y = 1:size(CTMconnect,2)
                        if CTMconnect(x,y) == row(ind(2))
                            CTMconnect(x,y) = row(ind(1));
                        end
                    end
                end
            end
        end

        %shift y-coordinate of vertices at intersecting face so they
connect
        for i = 1:size(CTMmesh,1)
            if CTMmesh(i,2) >= 0.8500 && CTMmesh(i,2) <= 1.0100
%intersecting plane for 4th monomer along y-axis
                if CTMmesh(i,3) >= -2.2286 && CTMmesh(i,3) <= -1.4886
%bounds of case 5 for initial loop - (radius + 0.8586)
                    CTMmesh(i,1) = CTMmesh(i,1)*pf; %can toy with if
i want to adjust pinch
                    CTMmesh(i,2) = 0.9293;
                    if CTMmesh(i,3) > moncenter(3)
                        CTMmesh(i,3) =
CTMmesh(i,3)+((CTMmesh(i,3)+(radius + 0.8586))*scale); %can toy with
if i want to adjust pinch
                    end
                    if CTMmesh(i,3) < moncenter(3)
                        CTMmesh(i,3) = CTMmesh(i,3)-
(abs(CTMmesh(i,3)+(radius + 0.8586))*scale); %can toy with if i want
to adjust pinch
                    end
                end
            end
        end

        %remove duplicate coordinates on y-plane of intersection
        [row,~] = find(CTMmesh(:,2) == 0.9293 & CTMmesh(:,3) <= -
1.3900); %look only on desired plane of intersection
```

```
        [correctRow,~] = find(CTMmesh(row,3) <= -1.3900 &
CTMmesh(row,3) >= -2.3300); %only look on desired particle interface
        intersect = CTMmesh(row(correctRow),:);
        for i = 1:size(intersect,1)
            if isnan(intersect(i,:))
                continue
            end
            tf = ismember(intersect,intersect(i,:),'rows');
            ind = find(tf == 1);
            if numel(ind) > 1
                CTMmesh(row(ind(2)),:) = NaN;
                intersect(ind(2),:) = NaN; %just for diagnostics
                %loop over connect array and renumber vertices
                for x = 1:size(CTMconnect,1)
                    for y = 1:size(CTMconnect,2)
                        if CTMconnect(x,y) == row(ind(2))
                            CTMconnect(x,y) = row(ind(1));
                        end
                    end
                end
            end
        end
    end

%%%If faces 2 and 4 were removed, adds monomer to complete 4-mer%%%
    if isequal(emptyFaces,[2,4]) || isequal(emptyFaces,[4,2])
        %fourth monomer should have face 2 removed
        new3Mesh = meshCoords;
        %removing face 2
        for i2 = 1:m
            if meshCoords(i2,1) < -0.8586 %on back face
                if meshCoords(i2,2) > -0.3659 && meshCoords(i2,2) <
0.3659
                    if meshCoords(i2,3) > -0.3591 && meshCoords(i2,3)
< 0.3591
                        new3Mesh(i2,:) = NaN;
                    end
                end
            end
        end

        %combine coordinates
        new3Mesh(:,1) = new3Mesh(:,1) + (radius + 0.8586)*2;
        CTMmesh = [CTMmesh ; zeros(m,n)];
        CTMmesh(1+((f+1)*m):((f+2)*m),:) = new3Mesh;
        CTMconnect = [CTMconnect; connect+size(newMesh,1)*(f+1)];

        %shift x-coordinate of vertices at intersecting face so they
connect
        for i = 1:size(CTMmesh,1)
            if CTMmesh(i,1) >= 2.717 && CTMmesh(i,1) <= 2.859
%intersecting plane for 4th monomer along y-axis
```

```
                if CTMmesh(i,3) >= -0.3700 && CTMmesh(i,3) <= 0.3700
&& CTMmesh(i,2) >= -0.3833 && CTMmesh(i,2) <= 0.3833
                    CTMmesh(i,1) = 2.788; %can toy with if i want to
adjust pinch
                    CTMmesh(i,2) = CTMmesh(i,2)*pf;
                    CTMmesh(i,3) = CTMmesh(i,3)*pf; %can toy with if
i want to adjust pinch
                end
            end
        end

        %remove duplicate coordinates on x-plane of intersection
        [row,~] = find(CTMmesh(:,1) == 2.788); %look only on desired
plane of intersection
        [correctRow,~] = find(CTMmesh(row,3) >= -0.3700*pf &
CTMmesh(row,3) <= 0.3700*pf & CTMmesh(row,2) >= -0.3833*pf &
CTMmesh(row,2) <= 0.3833*pf); %only look on desired particle
interface
        intersect = CTMmesh(row(correctRow),:);
        for i = 1:size(intersect,1)
            if isnan(intersect(i,:))
                continue
            end
            tf = ismember(intersect,intersect(i,:),'rows');
            ind = find(tf == 1);
            if numel(ind) > 1
                CTMmesh(row(ind(2)),:) = NaN;
                intersect(ind(2),:) = NaN; %just for diagnostics
                %loop over connect array and renumber vertices
                for x = 1:size(CTMconnect,1)
                    for y = 1:size(CTMconnect,2)
                        if CTMconnect(x,y) == row(ind(2))
                            CTMconnect(x,y) = row(ind(1));
                        end
                    end
                end
            end
        end
    end


%%%If faces 5 and 6 were removed, adds monomer to complete 4-mer%%%
    if isequal(emptyFaces,[5,6]) || isequal(emptyFaces,[6,5])
        %fourth monomer should have face 6 removed
        new3Mesh = meshCoords;
        %removing face 6
        for i6 = 1:m
            if meshCoords(i6,2) < -0.8586 %on right face
                if meshCoords(i6,1) > -0.3659 && meshCoords(i6,1) <
0.3659
                    if meshCoords(i6,3) > -0.3591 && meshCoords(i6,3)
< 0.3591
                        new3Mesh(i6,:) = NaN;
```

```
                end
            end
        end
    end

    %combine coordinates
    new3Mesh(:,2) = new3Mesh(:,2) + (radius + 0.8586)*2;
    CTMmesh = [CTMmesh ; zeros(m,n)];
    CTMmesh(1+((f+1)*m):((f+2)*m),:) = new3Mesh;
    CTMconnect = [CTMconnect; connect+size(newMesh,1)*(f+1)];

    %shift y-coordinate of vertices at intersecting face so they
connect
    for i = 1:size(CTMmesh,1)
        if CTMmesh(i,2) >= 2.717 && CTMmesh(i,2) <= 2.859
%intersecting plane for 4th monomer along y-axis
            if CTMmesh(i,3) >= -0.3700 && CTMmesh(i,3) <= 0.3700
&& CTMmesh(i,1) >= -0.3833 && CTMmesh(i,1) <= 0.3833
                CTMmesh(i,1) = CTMmesh(i,1)*pf; %can toy with if
i want to adjust pinch
                CTMmesh(i,2) = 2.788;
                CTMmesh(i,3) = CTMmesh(i,3)*pf; %can toy with if
i want to adjust pinch
            end
        end
    end

    %remove duplicate coordinates on y-plane of intersection
    [row,~] = find(CTMmesh(:,2) == 2.788); %look only on desired
plane of intersection
    [correctRow,~] = find(CTMmesh(row,3) >= -0.3700*pf &
CTMmesh(row,3) <= 0.3700*pf & CTMmesh(row,1) >= -0.3833*pf &
CTMmesh(row,1) <= 0.3833*pf); %only look on desired particle
interface
    intersect = CTMmesh(row(correctRow),:);
    for i = 1:size(intersect,1)
        if isnan(intersect(i,:))
            continue
        end
        tf = ismember(intersect,intersect(i,:),'rows');
        ind = find(tf == 1);
        if numel(ind) > 1
            CTMmesh(row(ind(2)),:) = NaN;
            intersect(ind(2),:) = NaN; %just for diagnostics
            %loop over connect array and renumber vertices
            for x = 1:size(CTMconnect,1)
                for y = 1:size(CTMconnect,2)
                    if CTMconnect(x,y) == row(ind(2))
                        CTMconnect(x,y) = row(ind(1));
                    end
                end
            end
        end
```

```
        end
    end

%%%Final Cleanup%%%
    %mark removed vertices in connect array
    for i = 1:size(CTMconnect,1)
        for j = 1:size(CTMconnect,2)
            if sum(isnan(CTMmesh(CTMconnect(i,j),:))) >= 1
                CTMconnect(i,j) = NaN;
            end
        end
    end

    %reorder node numbers in connect array
    i = 1;
    counter = 0;
    totcounter = 0;
    while i <= size(CTMmesh,1)
        if sum(isnan(CTMmesh(i,:))) >= 1
            counter = counter + 1;
        else
            if i~=1 && sum(isnan(CTMmesh(i-1,:))) >= 1
                for x = 1:size(CTMconnect,1)
                    for y = 1:size(CTMconnect,2)
                        if CTMconnect(x,y) >= i-totcounter;
                            CTMconnect(x,y) = CTMconnect(x,y) -
counter;
                        end
                    end
                end
                totcounter = totcounter + counter;
                counter = 0;
            end
        end
        i = i + 1;
    end

    %cleanup for vertices array
    r = size(CTMmesh,1);
    while r >= 1
        if sum(isnan(CTMmesh(r,:))) >= 1
            CTMmesh(r,:) = [];
        end
        r = r - 1;
    end

    %cleanup for connect array
    r = size(CTMconnect,1);
    while r >= 1
        if sum(isnan(CTMconnect(r,:))) >= 1
            CTMconnect(r,:) = [];
        end
```

```
        r = r - 1;
    end

%if no faces were removed from mesh (and it's still one cell)
else
    CTMmesh = newMesh;
    CTMconnect = connect;
end

%scaling of entire particle (for images,etc.
CTMmesh = CTMmesh.*6.9590;

%plot new particle
% connect3 =
[CTMconnect(:,1),CTMconnect(:,5),CTMconnect(:,2),CTMconnect(:,6),CTMc
onnect(:,3),CTMconnect(:,7),CTMconnect(:,4),CTMconnect(:,8)];
% figure;
% particle2 =
patch('faces',connect3,'vertices',CTMmesh,'FaceColor',[1 .5
.5],'EdgeColor',[1 0 0]);
% xlabel('X (\mum)','FontSize',14)
% ylabel('Y (\mum)','FontSize',14)
% zlabel('Z (\mum)','FontSize',14)
% title('4-mer-Tetrahedron')
% set(gca,'FontSize',14,'LineWidth',3)

%optional rotation if 4mer case 1->5->0 was selected, to orient
particle
%parallel to flow, with added monomers oriented in positive axes
% figure;
% h = plot3(CTMmesh(:,1),CTMmesh(:,2),CTMmesh(:,3),'.');
% rotate(h,[0 0 1],-90,[0 0 0])
% rotate(h,[0 1 0],-90,[0 0 0])
% x = h.XData;
% y = h.YData;
% z = h.ZData;
% CTMmesh = [x' y' z'];

%randomly reorient particle and plot
% newCTMmesh = meshRotate(CTMmesh);
% figure;
% particle3 =
patch('faces',connect3,'vertices',newCTMmesh,'FaceColor',[1 .5
.5],'EdgeColor',[1 0 0]);
% CTMmesh = newCTMmesh;

%dimension checks
maxx = max(CTMmesh(:,1));
minx = min(CTMmesh(:,1));
maxy = max(CTMmesh(:,2));
miny = min(CTMmesh(:,2));
maxz = max(CTMmesh(:,3));
```

```
minz = min(CTMmesh(:,3));

newCenter = [(maxx+minx)/2,(maxy+miny)/2,(maxz+minz)/2];
radiusx = abs(maxx-newCenter(1));
radiusy = abs(maxy-newCenter(2));
radiusz = abs(maxz-newCenter(3));
newRadius = [radiusx radiusy radiusz];

%save data to external files
% %connectivity matrix
% fid = fopen('File_name.dat','w');
% for t = 1:size(CTMconnect,1)
%   fprintf(fid,'%d ', CTMconnect(t,:));
%   fprintf(fid,'\n');
% end
% fclose(fid);
%
% %vertices matrix
% fid = fopen('File_name.dat','w');
% for t = 1:size(CTMmesh,1)
%   fprintf(fid,'%f ',CTMmesh(t,:));
%   fprintf(fid,'\n');
% end
% fclose(fid);


end
```

A.6 MATLAB CODE FOR VISUALIZATION OF MULTIPARTICLE ADHESIVE DYNAMICS SIMULATION

```
%This program simulates rotational motion of a 2mer CTM
%Edit(1) changes output so that time between each frame is even
%Edit(2) includes update to highlight elements where bonds have
formed
%Edit(3) adds instantaneous velocity and bond number plots

nel=4;

%Call meshAdjust program to determine faces and vertices of polygonal
%object
[CTMconnect, CTMmesh] = meshAdjust4(nel);
Xs = CTMmesh;
connect = CTMconnect;
connect2=[connect(:,1),connect(:,5),connect(:,2),connect(:,6),connect
(:,3),connect(:,7),connect(:,4),connect(:,8)];

pathdef = 'PATH';
suffix = '0';
%Importing CTM data
angles = load([pathdef,'CTMangles',suffix,'.txt']);
trans = load([pathdef,'CTMtrans',suffix,'.txt']);
trans0 = trans(1,1); %x-coordinate at start of interval
time = load([pathdef,'CTMrealtime',suffix,'.txt']);
bformed = load([pathdef,'CTMbformed',suffix,'.txt']);
bbroken = load([pathdef,'CTMbbroken',suffix,'.txt']);
bonds = load([pathdef,'CTMnumbondexist',suffix,'.txt']);
rec = length(trans(:,1));
rotatex=angles(:,1);
rotatey=angles(:,2);
rotatez=angles(:,3);
movex=trans(:,1);
movey=trans(:,2);
movez=trans(:,3);
zerotext = '0000';
p1 = 0;
j=1;
angleincrx(1) = rotatex(1);
angleincry(1) = rotatey(1);
angleincrz(1) = rotatez(1);
interval = 1.0e-4; %time interval between output frames
dt = 0; %delta time to be summed up between timesteps
count = 1; %counter for velocity points

%The bformed and bbroken vectors have the timestep of bond events in
the
```

```
%second column, but vectors are only as long as the number of bonds
in the
%simulation, so a counter is needed
fcounter = 1;
bcounter = 1;

bondnum = 0; %counter for number of bonds currently existing
formation = bformed(:,2);
breakage = bbroken(:,2);
bondcheck = zeros(size(bonds)); %output to see if calculated
instantaneous bond number is correct
bondelement = 0; %vector listing elements where bonds currently exist

CData = zeros(1,size(CTMconnect,1),3); %array containing RGB triplet
colors for each element
%standard element color is magenta, or 'm' short name
CData(:,:,1) = 1;
CData(:,:,2) = 0;
CData(:,:,3) = 1;

%diagnostic output to see which time steps output frames are being
made
fid = fopen('OutputSteps.txt','w');

for p = 2:rec
    angleincrx(p) = rotatex(p)-rotatex(p-1);
    angleincry(p) = rotatey(p)-rotatey(p-1);
    angleincrz(p) = rotatez(p)-rotatez(p-1);
end
Xsnew(:,:)=Xs(:,:);
for p = 1:rec
    for i = 1:size(Xsnew,1)
        if p >1
            x = Xsnew(i,1)-movex(p-1);
            y = Xsnew(i,2)-movey(p-1);
            z = Xsnew(i,3)-movez(p-1);
        else
            x = Xsnew(i,1);
            y = Xsnew(i,2);
            z = Xsnew(i,3);
        end
        rrot = sqrt(y^2+z^2);
        rotate = atan2(z,y)+angleincrx(p);
        y = rrot*cos(rotate);
        z = rrot*sin(rotate);
        Xsnew(i,1)=x;
        Xsnew(i,2)=y;
        Xsnew(i,3)=z;
    end
    for i = 1:size(Xsnew,1)
        x = Xsnew(i,1); y = Xsnew(i,2); z = Xsnew(i,3);
        rrot = sqrt(x^2+z^2);
```

```
        rotate = atan2(z,x)-angleincry(p);
        x = rrot*cos(rotate);
        z = rrot*sin(rotate);
        Xsnew(i,1)=x;
        Xsnew(i,2)=y;
        Xsnew(i,3)=z;
    end
    for i = 1:size(Xsnew,1)
        x = Xsnew(i,1);
        y = Xsnew(i,2);
        z = Xsnew(i,3);
        rrot = sqrt(x^2+y^2);
        rotate = atan2(y,x)+angleincrz(p);
        x = rrot*cos(rotate);
        y = rrot*sin(rotate);
        Xsnew(i,1)=x + movex(p);
        Xsnew(i,2)=y + movey(p);
        Xsnew(i,3)=z + movez(p);
    end

%Output frames are only made at timesteps of a set interval
    if p >=2
        %add up time between each step of time, which is every 100
steps
        dt = dt + time(p,2)-time(p-1,2);

        %p coincides with the output of time or trans, which is every
100
        %timesteps; evaluating bond formation and breakage however,
        %requires looking at every timestep
        for psub = time(p-1,1):time(p,1) %ex:between step 100-200
            %check for bond formation and breakage
            if psub <= formation(end)
                if psub == formation(fcounter)
                    %check formation to see if multiple bond events
occured at
                    %current timestep
                    if (fcounter ~= length(formation) &&
formation(fcounter + 1) == formation(fcounter))
                        fmultiples = find(formation ==
formation(fcounter));
                        %this array lists each element where a bond
has
                        %formed at the current step

bondelement(fcounter:fcounter+(length(fmultiples)-1)) =
bformed(fmultiples,1);
                        bondnum = bondnum + 1*length(fmultiples);
                        fcounter = fcounter + 1*length(fmultiples);
                    else
                        fsingle = find(formation ==
formation(fcounter));
                        bondelement(fcounter) = bformed(fsingle,1);
```

```
                        bondnum = bondnum + 1;
                        fcounter = fcounter + 1;
                    end
                end
            end
            if psub <=breakage(end)
                if psub == breakage(bcounter)
                    %check breakage to see if multiple bond events
occured at
                    %current timestep
                    if (bcounter ~= length(breakage) &&
breakage(bcounter + 1) == breakage(bcounter))
                        bmultiples = find(breakage ==
breakage(bcounter));
                        for bmsub = 1:length(bmultiples)
                            rembonds = find(bondelement ==
bbroken(bmultiples(bmsub),1),1,'first');
                            %when the bonds(s) is/are broken, just
change
                            %that element to NaN instead of removing
it so
                            %that fcounter is not thrown off
                            bondelement(rembonds) = NaN;
                        end
                        bondnum = bondnum - 1*length(bmultiples);
                        bcounter = bcounter + 1*length(bmultiples);
                    else
                        bsingle = find(breakage ==
breakage(bcounter));
                        rembond = find(bondelement ==
bbroken(bsingle,1),1,'first');
                        bondelement(rembond) = NaN;
                        bondnum = bondnum - 1;
                        bcounter = bcounter + 1;
                    end
                end
            end
        end

        %check to see if bondnum output is correct
        bondcheck(p) = bondnum;

        if dt >= interval
            %diagnostic output to see which timesteps frames are
being made
            fprintf(fid,'\t%d', time(p,1));
            fprintf(fid,'\t');
            fprintf(fid,'%f', time(p,2));
            fprintf(fid,'\n');

            %first calculate the velocity
            velocity(count) = (trans(p,1)-trans0)/dt;
            vtime(count,:) = time(p,:);
```

```
            trans0 = trans(p);

            vbonds(count) = bondcheck(p); %instantaneous bond number
array the same size as vtime

            %next several loops will fade elements that had bonds,
            %successively fading the color from yellow (bond exists)
back
            %to magenta (no bonds)

            %if element(s) were the faded color, return them to the
default color (magenta)
            for eleold = 1:size(CTMconnect,1)
                if CData(:,eleold,1) == 1
                    if CData(:,eleold,2) == 0.5
                        if CData(:,eleold,3) == 0.5
                            CData(:,eleold,1) = 1;
                            CData(:,eleold,2) = 0;
                            CData(:,eleold,3) = 1;
                        end
                    end
                end
            end

            %if element(s) were intermediate color, change to a more
faded
            %color
            for ele = 1:size(CTMconnect,1)
                if CData(:,ele,1) == 1
                    if CData(:,ele,2) == 0.75
                        if CData(:,ele,3) == 0.25
                            CData(:,ele,1) = 1;
                            CData(:,ele,2) = 0.5;
                            CData(:,ele,3) = 0.5;
                        end
                    end
                end
            end
            %if bond(s) previously existed, and element was yellow,
change it to an intermediate color
            for ele = 1:size(CTMconnect,1)
                if CData(:,ele,1) == 1
                    if CData(:,ele,2) == 1
                        if CData(:,ele,3) == 0
                            CData(:,ele,1) = 1;
                            CData(:,ele,2) = 0.75;
                            CData(:,ele,3) = 0.25;
                        end
                    end
                end
            end
```

```
            %determine which, if any, elements have bonds existing,
then
            %color elements based on bond status
            if bondnum > 0
                existingBondsCheck = ~isnan(bondelement);
                existingBonds =
unique(bondelement(existingBondsCheck));
                %color elements yellow where bonds now exist
                CData(:,existingBonds,1) = 1;
                CData(:,existingBonds,2) = 1;
                CData(:,existingBonds,3) = 0;
            end

            %Subfigure: Side View
            p1 = p1+1;
            framename = ['PATH\Movie Files/movieframe',zerotext(1:4-
length(num2str(p1))),num2str(p1),'.jpg'];
            clf
            subplot(2,2,1)
            grid on
            axis([movex(p)-2 movex(p)+2 movey(p)-2 movey(p)+2 0 2])

patch('faces',connect2,'vertices',Xsnew(:,:),'FaceColor','flat','CDat
a',CData,'EdgeColor','k')
            xlabel('X (\mum)','FontSize',12)
            zlabel('Z (\mum)','FontSize',12)
            title('Side View','FontSize',12)
            set(gca,'FontSize',10,'LineWidth',1)
            axis equal
            view(0,0)



            %Subfigure: Back View
            subplot(2,2,2)
            grid on
            axis([movex(p)-2 movex(p)+2 movey(p)-2 movey(p)+2 0 2])

patch('faces',connect2,'vertices',Xsnew(:,:),'FaceColor','flat','CDat
a',CData,'EdgeColor','k')
            ylabel('Y (\mum)','FontSize',12)
            zlabel('Z (\mum)','FontSize',12)
            title('Back View','FontSize',12)
            set(gca,'FontSize',10,'LineWidth',1)
            axis equal
             view(-90,0)

            %Subfigure: Instantaneous Velocity
            subplot(2,3,4)
            plot(vtime(:,2),velocity','LineWidth',2);
            ylabel('Velocity
(\mum/s)','FontSize',12,'VerticalAlignment','bottom')
```

```
            xlabel('Time
(s)','FontSize',12,'VerticalAlignment','cap')
            title('Instantaneous Velocity','FontSize',12)
            set(gca,'FontSize',10,'LineWidth',1)
            xlim([vtime(count,2)-(5*interval)
vtime(count,2)+(5*interval)])

            %Subfigure: Bottom-front view (Best for viewing bond
formation)
            subplot(2,3,5)
            axis([movex(p)-2 movex(p)+2 movey(p)-2 movey(p)+2 0 2])

patch('faces',connect2,'vertices',Xsnew(:,:),'FaceColor','flat','CDat
a',CData,'EdgeColor','k')
            axis equal
            xlabel('X (\mum)','FontSize',12)
            ylabel('Y (\mum)','FontSize',12)
            zlabel('Z (\mum)','FontSize',12)
            title('Bottom-Front View','FontSize',12)
            view(-230,-30)
            set(gca,'FontSize',10,'LineWidth',1)
                grid on

            %Subfigure: Instantaneous Bond Number
            subplot(2,3,6)
            plot(vtime(:,2)',vbonds,'LineWidth',2);
            ylabel('Bond
Number','FontSize',12,'VerticalAlignment','bottom')
            xlabel('Time
(s)','FontSize',12,'VerticalAlignment','cap')
            title('Instantaneous Bond Number','FontSize',12)
            axis([vtime(count,2)-(10*interval)
vtime(count,2)+(10*interval) vbonds(count)-5 vbonds(count)+5])
            set(gca,'FontSize',10,'LineWidth',1)

            set(gcf, 'Position', get(0,'Screensize')); % Maximize
figure
            print(gcf,'-djpeg100',framename)

            dt = 0;
            count = count + 1;
        end
    end
end
fclose(fid);
```

REFERENCES

1. Stewart, B.W. and C.P. Wild. *World cancer report 2014*. 2014; Available from: https://publications.cancerresearchuk.org/downloads/product/CS_REPORT_WORLD.pdf.
2. Hanahan, D. and Robert A. Weinberg, *Hallmarks of Cancer: The Next Generation.* Cell, 2011. **144**(5): p. 646-674.
3. Leber, M.F. and T. Efferth, *Molecular principles of cancer invasion and metastasis (review).* Int J Oncol, 2009. **34**(4): p. 881-95.
4. Chiang, A.C. and J. Massagué, *Molecular Basis of Metastasis.* New England Journal of Medicine, 2008. **359**(26): p. 2814-2823.
5. Paterlini-Brechot, P. and N.L. Benali, *Circulating tumor cells (CTC) detection: Clinical impact and future directions.* Cancer Letters, 2007. **253**(2): p. 180-204.
6. Shayan, R., M.G. Achen, and S.A. Stacker, *Lymphatic vessels in cancer metastasis: bridging the gaps.* Carcinogenesis, 2006. **27**(9): p. 1729-1738.
7. Chang, Y.S., et al., *Mosaic blood vessels in tumors: Frequency of cancer cells in contact with flowing blood.* Proceedings of the National Academy of Sciences, 2000. **97**(26): p. 14608-14613.
8. Howlader, N., et al., *SEER Cancer Statistics Review, 1975-2013*. 2016, Bethesda, MD: National Cancer Institute.

9. *Colorectal Cancer Facts & Figures 2014-2016*. 2014, Atlanta, GA: American Cancer Society.

10. Crea, F., et al., *Epigenetics and chemoresistance in colorectal cancer: An opportunity for treatment tailoring and novel therapeutic strategies.* Drug Resistance Updates, 2011. **14**(6): p. 280-296.

11. Holst, S., M. Wuhrer, and Y. Rombouts, *Chapter Six - Glycosylation Characteristics of Colorectal Cancer*, in *Advances in Cancer Research*, R.D. Richard and E.B. Lauren, Editors. 2015, Academic Press. p. 203-256.

12. Bendas, G. and L. Borsig, *Cancer Cell Adhesion and Metastasis: Selectins, Integrins, and the Inhibitory Potential of Heparins.* International Journal of Cell Biology, 2012. **2012**: p. 10.

13. Merten, M. and P. Thiagarajan, *P-Selectin Expression on Platelets Determines Size and Stability of Platelet Aggregates.* Circulation, 2000. **102**(16): p. 1931-1936.

14. Ley, K., *The role of selectins in inflammation and disease.* Trends in Molecular Medicine, 2003. **9**(6): p. 263-268.

15. Setiadi, H. and R.P. McEver, *Clustering endothelial E-selectin in clathrin-coated pits and lipid rafts enhances leukocyte adhesion under flow.* Blood, 2008. **111**(4): p. 1989-1998.

16. Huang, R.B. and O. Eniola-Adefeso, *Shear Stress Modulation of IL-1β-Induced E-Selectin Expression in Human Endothelial Cells.* PLoS ONE, 2012. **7**(2): p. e31874.

17. Varki, A., *Selectin ligands.* Proceedings of the National Academy of Sciences of the United States of America, 1994. **91**(16): p. 7390-7397.

18. Luthur Siu-Lun, C., et al., *Biophysics of selectin–ligand interactions in inflammation and cancer.* Physical Biology, 2011. **8**(1): p. 015013.

19. Barthel, S.R., et al., *Targeting selectins and selectin ligands in inflammation and cancer.* Expert Opinion on Therapeutic Targets, 2007. **11**(11): p. 1473-1491.

20. Zarbock, A., et al., *Leukocyte ligands for endothelial selectins: specialized glycoconjugates that mediate rolling and signaling under flow.* Blood, 2011. **118**(26): p. 6743-6751.

21. Tomlinson, J., et al., *Human colon cancer cells express multiple glycoprotein ligands for E-selectin.* Int J Oncol, 2000. **16**(2): p. 347-53.

22. Mannori, G., et al., *Differential colon cancer cell adhesion to E-, P-, and L-selectin: role of mucin-type glycoproteins.* Cancer Res, 1995. **55**(19): p. 4425-31.

23. McEver, R.P. and C. Zhu, *Rolling cell adhesion.* Annu Rev Cell Dev Biol, 2010. **26**: p. 363-96.

24. Shen, B., M.K. Delaney, and X. Du, *Inside-out, outside-in, and inside-outside-in: G protein signaling in integrin-mediated cell adhesion, spreading, and retraction.* Current opinion in cell biology, 2012. **24**(5): p. 600-606.

25. Tandon, P. and S.L. Diamond, *Kinetics of β2-Integrin and L-Selectin Bonding during Neutrophil Aggregation in Shear Flow.* Biophysical Journal, 1998.

75(6): p. 3163-3178.

26. Yoshida, M., et al., *Phosphorylation of the Cytoplasmic Domain of E-Selectin Is Regulated During Leukocyte-Endothelial Adhesion.* The Journal of Immunology, 1998. **161**(2): p. 933-941.

27. Hu, Y., et al., *E-Selectin-Dependent Signaling Via the Mitogen-Activated Protein Kinase Pathway in Vascular Endothelial Cells.* The Journal of Immunology, 2000. **165**(4): p. 2142-2148.

28. Fernandez, S., et al., *TP53 mutations detected in circulating tumor cells present in the blood of metastatic triple negative breast cancer patients.* Breast Cancer Research, 2014. **16**(5): p. 445.

29. Hou, J.-M., et al., *Circulating Tumor Cells as a Window on Metastasis Biology in Lung Cancer.* The American Journal of Pathology, 2011. **178**(3): p. 989-996.

30. Glinsky, V.V., et al., *Intravascular Metastatic Cancer Cell Homotypic Aggregation at the Sites of Primary Attachment to the Endothelium.* Cancer Research, 2003. **63**(13): p. 3805-3811.

31. Al-Mehdi, A.B., et al., *Intravascular origin of metastasis from the proliferation of endothelium-attached tumor cells: a new model for metastasis.* Nat Med, 2000. **6**(1): p. 100-102.

32. Fortuna-Costa, A., et al., *Extracellular Galectin-3 in Tumor Progression and Metastasis.* Frontiers in Oncology, 2014. **4**: p. 138.

33. Porquet, N. and J. Huot, *Signal Transduction in Tumor-Endothelial Cell*

Communication, in *Liver Metastasis: Biology and Clinical Management*, P. Brodt, Editor. 2011, Springer Netherlands. p. 187-212.

34. Mitchell, M.J. and M.R. King, *Physical Biology in Cancer. 3. The role of cell glycocalyx in vascular transport of circulating tumor cells*. Vol. 306. 2014. C89-C97.

35. Luzzi, K.J., et al., *Multistep Nature of Metastatic Inefficiency.* The American Journal of Pathology, 1998. **153**(3): p. 865-873.

36. Wong, C.W., et al., *Apoptosis: An Early Event in Metastatic Inefficiency.* Cancer Research, 2001. **61**(1): p. 333-338.

37. Fidler, I.J., D.M. Gersten, and C.W. Riggs, *Relationship of host immune status to tumor cell arrest, distribution, and survival in experimental metastasis.* Cancer, 1977. **40**(1): p. 46-55.

38. *Cancer Facts & Figures 2016*. 2016, Atlanta, GA: American Cancer Society.

39. Mehlen, P. and A. Puisieux, *Metastasis: a question of life or death.* Nat Rev Cancer, 2006. **6**(6): p. 449-458.

40. Bersini, S., et al., *In vitro models of the metastatic cascade: from local invasion to extravasation.* Drug Discovery Today, 2014. **19**(6): p. 735-742.

41. Minn, A.J., et al., *Genes that mediate breast cancer metastasis to lung.* Nature, 2005. **436**(7050): p. 518-524.

42. Bos, P.D., et al., *Genes that mediate breast cancer metastasis to the brain.* Nature, 2009. **459**(7249): p. 1005-1009.

43. Kang, Y., et al., *A multigenic program mediating breast cancer metastasis to*

bone. Cancer Cell, 2003. **3**(6): p. 537-549.

44. Hsu, Y.L., et al., *Breast tumor-associated osteoblast-derived CXCL5 increases cancer progression by ERK/MSK1/Elk-1/Snail signaling pathway.* Oncogene, 2013. **32**(37): p. 4436-4447.

45. Claffey, K.P., et al., *Expression of Vascular Permeability Factor/Vascular Endothelial Growth Factor by Melanoma Cells Increases Tumor Growth, Angiogenesis, and Experimental Metastasis.* Cancer Research, 1996. **56**(1): p. 172-181.

46. Shibue, T. and R.A. Weinberg, *Integrin β1-focal adhesion kinase signaling directs the proliferation of metastatic cancer cells disseminated in the lungs.* Proceedings of the National Academy of Sciences, 2009. **106**(25): p. 10290-10295.

47. Urosevic, J., et al., *Colon cancer cells colonize the lung from established liver metastases through p38 MAPK signalling and PTHLH.* Nat Cell Biol, 2014. **16**(7): p. 685-694.

48. Abt, M.A., et al., *Evaluation of Lung Metastasis in Mouse Mammary Tumor Models by Quantitative Real-time PCR.* Journal of Visualized Experiments : JoVE, 2016(107): p. 10.3791/53329.

49. Ng, J., Y. Shin, and S. Chung, *Microfluidic platforms for the study of cancer metastasis.* Biomedical Engineering Letters, 2012. **2**(2): p. 72-77.

50. Labow, M.A., et al., *Characterization of E-selectin-deficient mice: Demonstration of overlapping function of the endothelial selectins.* Immunity,

1994. **1**(8): p. 709-720.

51. Dong, Z.M., et al., *The combined role of P- and E-selectins in atherosclerosis.* Journal of Clinical Investigation, 1998. **102**(1): p. 145-152.

52. Mayadas, T.N., et al., *Leukocyte rolling and extravasation are severely compromised in P selectin-deficient mice.* Cell, 1993. **74**(3): p. 541-554.

53. Frenette, P.S., et al., *Susceptibility to Infection and Altered Hematopoiesis in Mice Deficient in Both P- and E-Selectins.* Cell, 1996. **84**(4): p. 563-574.

54. Finger, E.B., et al., *Adhesion through L-selectin requires a threshold hydrodynamic shear.* Nature, 1996. **379**(6562): p. 266-269.

55. Goetz, D.J., et al., *Dynamics of neutrophil rolling over stimulated endothelium in vitro.* Biophysical Journal, 1994. **66**(6): p. 2202-2209.

56. Lawrence, M.B. and T.A. Springer, *Leukocytes roll on a selectin at physiologic flow rates: Distinction from and prerequisite for adhesion through integrins.* Cell, 1991. **65**(5): p. 859-873.

57. Smith, M.J., E.L. Berg, and M.B. Lawrence, *A Direct Comparison of Selectin-Mediated Transient, Adhesive Events Using High Temporal Resolution.* Biophysical Journal, 1999. **77**(6): p. 3371-3383.

58. Chen, S. and T.A. Springer, *An Automatic Braking System That Stabilizes Leukocyte Rolling by an Increase in Selectin Bond Number with Shear.* The Journal of Cell Biology, 1999. **144**(1): p. 185-200.

59. Yago, T., et al., *Catch bonds govern adhesion through L-selectin at threshold shear.* The Journal of Cell Biology, 2004. **166**(6): p. 913-923.

60. Abbassi, O., et al., *E-selectin supports neutrophil rolling in vitro under conditions of flow.* Journal of Clinical Investigation, 1993. **92**(6): p. 2719-2730.

61. Park, E.Y.H., et al., *Comparison of PSGL-1 Microbead and Neutrophil Rolling: Microvillus Elongation Stabilizes P-Selectin Bond Clusters.* Biophysical Journal, 2002. **82**(4): p. 1835-1847.

62. Goetz, D.J., et al., *Isolated P-selectin Glycoprotein Ligand-1 Dynamic Adhesion to P- and E-selectin.* The Journal of Cell Biology, 1997. **137**(2): p. 509-519.

63. Brunk, D.K., D.J. Goetz, and D.A. Hammer, *Sialyl Lewis(x)/E-selectin-mediated rolling in a cell-free system.* Biophysical Journal, 1996. **71**(5): p. 2902-2907.

64. Marshall, B.T., et al., *Direct observation of catch bonds involving cell-adhesion molecules.* Nature, 2003. **423**(6936): p. 190-193.

65. Bell, G.I., *Models for the Specific Adhesion of Cells to Cells.* Science, 1978. **200**(4342): p. 618-627.

66. Wayman, A.M., et al., *Triphasic Force Dependence of E-Selectin/Ligand Dissociation Governs Cell Rolling under Flow.* Biophysical Journal, 2010. **99**(4): p. 1166-1174.

67. Lipowsky, H.H., D. Riedel, and G.S. Shi, *In vivo mechanical properties of leukocytes during adhesion to venular endothelium.* Biorheology, 1991. **28**(1-2): p. 53-64.

68. Rodgers, S.D., R.T. Camphausen, and D.A. Hammer, *Sialyl LewisX-Mediated, PSGL-1-Independent Rolling Adhesion on P-selectin.* Biophysical Journal, 2000. **79**(2): p. 694-706.

69. King, M.R., et al., *A physical sciences network characterization of circulating tumor cell aggregate transport.* American Journal of Physiology - Cell Physiology, 2015. **308**(10): p. C792-C802.

70. Rana, K., J.L. Liesveld, and M.R. King, *Delivery of apoptotic signal to rolling cancer cells: A novel biomimetic technique using immobilized TRAIL and E-selectin.* Biotechnology and Bioengineering, 2009. **102**(6): p. 1692-1702.

71. Napier, S.L., et al., *Selectin Ligand Expression Regulates the Initial Vascular Interactions of Colon Carcinoma Cells: THE ROLES OF CD44V AND ALTERNATIVE SIALOFUCOSYLATED SELECTIN LIGANDS.* Journal of Biological Chemistry, 2007. **282**(6): p. 3433-3441.

72. Tremblay, P.-L., J. Huot, and F.A. Auger, *Mechanisms by which E-Selectin Regulates Diapedesis of Colon Cancer Cells under Flow Conditions.* Cancer Research, 2008. **68**(13): p. 5167-5176.

73. Burdick, M.M. and K. Konstantopoulos, *Platelet-induced enhancement of LS174T colon carcinoma and THP-1 monocytoid cell adhesion to vascular endothelium under flow.* American Journal of Physiology - Cell Physiology, 2004. **287**(2): p. C539-C547.

74. Thomas, S.N., et al., *Carcinoembryonic Antigen and CD44 Variant Isoforms Cooperate to Mediate Colon Carcinoma Cell Adhesion to E- and L-selectin in*

Shear Flow. Journal of Biological Chemistry, 2008. **283**(23): p. 15647-15655.

75. Gong, L., et al., *P-selectin-mediated platelet activation promotes adhesion of non-small cell lung carcinoma cells on vascular endothelial cells under flow.* Mol Med Rep, 2012. **5**(4): p. 935-42.

76. Springer, T.A., *Traffic Signals on Endothelium for Lymphocyte Recirculation and Leukocyte Emigration.* Annual Review of Physiology, 1995. **57**(1): p. 827-872.

77. Chang, K.C. and D.A. Hammer, *Adhesive dynamics simulations of sialyl-Lewis(x)/E-selectin-mediated rolling in a cell-free system.* Biophys J, 2000. **79**(4): p. 1891-902.

78. Hammer, D.A. and S.M. Apte, *Simulation of cell rolling and adhesion on surfaces in shear flow: general results and analysis of selectin-mediated neutrophil adhesion.* Biophysical Journal, 1992. **63**(1): p. 35-57.

79. Chang, K.-C., F.J.T. David, and D.A. Hammer, *The State Diagram for Cell Adhesion Under Flow: Leukocyte Rolling and Firm Adhesion.* Proceedings of the National Academy of Sciences of the United States of America, 2000. **97**(21): p. 11262-11267.

80. Mitchell, M.J., et al., *TRAIL-coated leukocytes that kill cancer cells in the circulation.* Proceedings of the National Academy of Sciences, 2014. **111**(3): p. 930-935.

81. Gakhar, G., et al., *Circulating Tumor Cells from Prostate Cancer Patients Interact with E-Selectin under Physiologic Blood Flow.* PLoS ONE, 2013.

8(12): p. e85143.

82. Hughes, A.D., et al., *Microtube Device for Selectin-Mediated Capture of Viable Circulating Tumor Cells from Blood.* Clinical Chemistry, 2012. **58**(5): p. 846-853.

83. Oh, J., et al., *Analytical cell adhesion chromatography reveals impaired persistence of metastatic cell rolling adhesion to P-selectin.* Journal of Cell Science, 2015. **128**(20): p. 3731-3743.

84. Au, S.H., et al., *Clusters of circulating tumor cells traverse capillary-sized vessels.* Proceedings of the National Academy of Sciences, 2016. **113**(18): p. 4947-4952.

85. Sarioglu, A.F., et al., *A microfluidic device for label-free, physical capture of circulating tumor cell clusters.* Nat Meth, 2015. **12**(7): p. 685-691.

86. Phillips, K.G., et al., *The thrombotic potential of circulating tumor microemboli: computational modeling of circulating tumor cell-induced coagulation.* American Journal of Physiology - Cell Physiology, 2015. **308**(3): p. C229-C236.

87. Lee, A.M., et al., *Modeling and Simulation of Procoagulant Circulating Tumor Cells in Flow.* Frontiers in Oncology, 2012. **2**: p. 108.

88. Puliafito, A., et al., *Three-dimensional chemotaxis-driven aggregation of tumor cells.* Scientific Reports, 2015. **5**: p. 15205.

89. Rejniak, K.A., *Circulating Tumor Cells: When a Solid Tumor Meets a Fluid Microenvironment.* Advances in experimental medicine and biology, 2016.

936: p. 93-106.

90. Lee, D., et al., *Adhesive dynamics simulations of the mechanical shedding of L-selectin from the neutrophil surface.* J Theor Biol, 2009. **260**(1): p. 27-30.

91. King, M.R. and D.A. Hammer, *Multiparticle Adhesive Dynamics. Interactions between Stably Rolling Cells.* Biophysical Journal, 2001. **81**(2): p. 799-813.

92. King, M.R. and D.A. Hammer, *Multiparticle adhesive dynamics: Hydrodynamic recruitment of rolling leukocytes.* Proceedings of the National Academy of Sciences of the United States of America, 2001. **98**(26): p. 14919-14924.

93. Mody, N.A. and M.R. King, *Platelet Adhesive Dynamics. Part I: Characterization of Platelet Hydrodynamic Collisions and Wall Effects*. Biophysical Journal, 2008. **95**(5): p. 2539-2555.

94. Mody, N.A. and M.R. King, *Platelet Adhesive Dynamics. Part II: High Shear-Induced Transient Aggregation via GPIbα-vWF-GPIbα Bridging.* Biophysical Journal, 2008. **95**(5): p. 2556-2574.

95. Kim, S. and S.J. Karrila, *The Boundary Integral Equations for Stokes Flow*. Microhydrodynamics: Principles and Selected Applications, ed. H. Brenner. 1991, Stoneham, MA: Butterworth-Heinemann.

96. Jeffery, G.B., *On the Steady Rotation of a Solid of Revolution in a Viscous Fluid.* Proceedings of the London Mathematical Society, 1915. **s2_14**(1): p. 327-338.

97. Goldman, A.J., R.G. Cox, and H. Brenner, *Slow viscous motion of a sphere*

parallel to a plane wall—I Motion through a quiescent fluid. Chemical Engineering Science, 1967. **22**(4): p. 637-651.

98. Goldman, A.J., R.G. Cox, and H. Brenner, *Slow viscous motion of a sphere parallel to a plane wall—II Couette flow.* Chemical Engineering Science, 1967. **22**(4): p. 653-660.

99. Bell, G.I., M. Dembo, and P. Bongrand, *Cell adhesion. Competition between nonspecific repulsion and specific bonding.* Biophysical Journal, 1984. **45**(6): p. 1051-1064.

100. Jayageeth, C., V.I. Sharma, and A. Singh, *Dynamics of short fiber suspensions in bounded shear flow.* International Journal of Multiphase Flow, 2009. **35**(3): p. 261-269.

101. Wang, W., N.A. Mody, and M.R. King, *Multiscale model of platelet translocation and collision.* Journal of Computational Physics, 2013. **244**(0): p. 223-235.

102. Chang, K.-C. and D.A. Hammer, *The Forward Rate of Binding of Surface-Tethered Reactants: Effect of Relative Motion between Two Surfaces.* Biophysical Journal, 1999. **76**(3): p. 1280-1292.

103. Power, H. and G. Miranda, *Second Kind Integral Equation Formulation of Stokes' Flows Past a Particle of Arbitrary Shape.* SIAM Journal on Applied Mathematics, 1987. **47**(4): p. 689-698.

104. Karrila, S.J., Y.O. Fuentes, and S. Kim, *Parallel Computational Strategies for Hydrodynamic Interactions Between Rigid Particles of Arbitrary Shape in a*

Viscous Fluid. Journal of Rheology, 1989. **33**(6): p. 913-947.

105. Hickey, M.J., et al., *Varying Roles of E-Selectin and P-Selectin in Different Microvascular Beds in Response to Antigen.* The Journal of Immunology, 1999. **162**(2): p. 1137-1143.

106. Sawada, R., S. Tsuboi, and M. Fukuda, *Differential E-selectin-dependent adhesion efficiency in sublines of a human colon cancer exhibiting distinct metastatic potentials.* Journal of Biological Chemistry, 1994. **269**(2): p. 1425-31.

107. Laferrière, J., F. Houle, and J. Huot, *Regulation of the Metastatic Process by E-Selectin and Stress-Activated Protein Kinase-2/p38.* Annals of the New York Academy of Sciences, 2002. **973**(1): p. 562-572.

108. Hanley, W.D., D. Wirtz, and K. Konstantopoulos, *Distinct kinetic and mechanical properties govern selectin-leukocyte interactions.* Journal of Cell Science, 2004. **117**(12): p. 2503-2511.

109. Alon, R., et al., *The Kinetics of L-selectin Tethers and the Mechanics of Selectin-mediated Rolling.* The Journal of Cell Biology, 1997. **138**(5): p. 1169-1180.

110. Massena, S. and M. Phillipson, *Intravascular Leukocyte Chemotaxis: The Rules of Attraction*. Hematology - Science and Practice, ed. C.H. Lawrie. 2012, Rijeka, Croatia: InTech.

111. Friedlander, T.W., G. Premasekharan, and P.L. Paris, *Looking back, to the future of circulating tumor cells.* Pharmacology & Therapeutics, 2014. **142**(3):

p. 271-280.

112. Vincent, Z., et al., *CD133-positive cancer stem cells from Colo205 human colon adenocarcinoma cell line show resistance to chemotherapy and display a specific metabolomic profile.* Genes & Cancer, 2014. **5**(7-8): p. 250-260.

113. Kuo, C.-T., C. Chang, and W.-S. Lee, *Folic acid inhibits COLO-205 colon cancer cell proliferation through activating the FRα/c-SRC/ERK1/2/NFκB/TP53 pathway: in vitro and in vivo studies.* Scientific Reports, 2015. **5**: p. 11187.

114. Long, M., et al., *Kinetic Measurements of Cell Surface E-Selectin/Carbohydrate Ligand Interactions.* Annals of Biomedical Engineering, 2001. **29**(11): p. 935-946.

115. Somers, W.S., et al., *Insights into the Molecular Basis of Leukocyte Tethering and Rolling Revealed by Structures of P- and E-Selectin Bound to SLe^X and PSGL-1.* Cell, 2000. **103**(3): p. 467-479.

116. Piper, J.W., R.A. Swerlick, and C. Zhu, *Determining force dependence of two-dimensional receptor-ligand binding affinity by centrifugation.* Biophysical Journal, 1998. **74**(1): p. 492-513.

117. Dembo, M., et al., *The Reaction-Limited Kinetics of Membrane-to-Surface Adhesion and Detachment.* Proceedings of the Royal Society of London. Series B, Biological Sciences, 1988. **234**(1274): p. 55-83.

118. Jeffery, G.B., *The Motion of Ellipsoidal Particles Immersed in a Viscous Fluid.* Proceedings of the Royal Society of London. Series A, 1922. **102**(715):

p. 161-179.

119. Nir, A. and A. Acrivos, *On the creeping motion of two arbitrary-sized touching spheres in a linear shear field.* Journal of Fluid Mechanics, 1973. **59**(02): p. 209-223.

120. Ahmadi, G. and J.B. McLaughlin, *Transport, Deposition, and Removal of Fine Particles: Biomedical Applications*. Medical Applications of Colloids, ed. E. Matijević. 2008, New York, NY: Springer Science & Business Media.

121. Mody, N.A. and M.R. King, *Three-dimensional simulations of a platelet-shaped spheroid near a wall in shear flow.* Physics of Fluids, 2005. **17**(11): p. 113302.

122. Gavze, E. and M. Shapiro, *Particles in a shear flow near a solid wall: Effect of nonsphericity on forces and velocities.* International Journal of Multiphase Flow, 1997. **23**(1): p. 155-182.

123. Harris, J.B. and J.F.T. Pittman, *Equivalent ellipsoidal axis ratios of slender rod-like particles.* Journal of Colloid and Interface Science, 1975. **50**(2): p. 280-282.

124. Leith, D., *Drag on Nonspherical Objects.* Aerosol Science and Technology, 1987. **6**(2): p. 153-161.

125. Mitchell, M.J. and M.R. King, *Physical Biology in Cancer. 3. The role of cell glycocalyx in vascular transport of circulating tumor cells.* American Journal of Physiology - Cell Physiology, 2014. **306**(2): p. C89-C97.

126. Jorge, N., et al., *High-definition imaging of circulating tumor cells and*

associated cellular events in non-small cell lung cancer patients: a longitudinal analysis. Physical Biology, 2012. **9**(1): p. 016004.

127. Mohamed, H., et al., *Isolation of tumor cells using size and deformation.* Journal of Chromatography A, 2009. **1216**(47): p. 8289-8295.

128. Marrinucci, D., et al., *Cytomorphology of Circulating Colorectal Tumor Cells: A Small Case Series.* Journal of Oncology, 2010. **2010**.

129. Marth, W., S. Aland, and A. Voigt, *Margination of white blood cells: a computational approach by a hydrodynamic phase field model.* Journal of Fluid Mechanics, 2016. **790**: p. 389-406.

130. Lee, A.M., *Modeling and simulation of circulating tumor cells in flow*, in *Aerospace Engineering*. 2014, University of Southern California: Los Angeles, California.

131. Aghaamoo, M., et al., *Deformability-based circulating tumor cell separation with conical-shaped microfilters: Concept, optimization, and design criteria.* Biomicrofluidics, 2015. **9**(3): p. 034106.

132. Tözeren, A. and K. Ley, *How do selectins mediate leukocyte rolling in venules?* Biophysical Journal, 1992. **63**(3): p. 700-709.

133. Skjetne, P., R.F. Ross, and D.J. Klingenberg, *Simulation of single fiber dynamics.* The Journal of Chemical Physics, 1997. **107**(6): p. 2108-2121.

134. Leal, L.G., *Creeping Flows – Three-Dimensional Problems*. Advanced Transport Phenomena: Fluid Mechanics and Convective Transport Processes, ed. A. Varma. 2006, Cambridge: Cambridge University Press.

135. Stover, C.A. and C. Cohen, *The motion of rodlike particles in the pressure-driven flow between two flat plates.* Rheologica Acta, 1990. **29**(3): p. 192-203.

136. Rocheleau, A.D., W. Wang, and M.R. King, *Effect of Pseudopod Extensions on Neutrophil Hemodynamic Transport Near a Wall.* Cellular and Molecular Bioengineering, 2016. **9**(1): p. 85-95.

137. Allard, W.J., et al., *Tumor Cells Circulate in the Peripheral Blood of All Major Carcinomas but not in Healthy Subjects or Patients With Nonmalignant Diseases.* Clinical Cancer Research, 2004. **10**(20): p. 6897-6904.

138. Yu, M., et al., *Circulating Breast Tumor Cells Exhibit Dynamic Changes in Epithelial and Mesenchymal Composition.* Science, 2013. **339**(6119): p. 580-584.

139. Sequist, L.V., et al., *The CTC-Chip: An Exciting New Tool to Detect Circulating Tumor Cells in Lung Cancer Patients.* Journal of Thoracic Oncology, 2009. **4**(3): p. 281-283.

140. Dena, M., et al., *Fluid biopsy in patients with metastatic prostate, pancreatic and breast cancers.* Physical Biology, 2012. **9**(1): p. 016003.

141. Foty, R., *A Simple Hanging Drop Cell Culture Protocol for Generation of 3D Spheroids.* Journal of Visualized Experiments : JoVE, 2011(51): p. 2720.

142. Bhatia, S.K., et al., *Rolling Adhesion Kinematics of Yeast Engineered To Express Selectins.* Biotechnology Progress, 2003. **19**(3): p. 1033-1037.

143. Woelke, A.L., et al., *Understanding Selectin Counter-Receptor Binding from Electrostatic Energy Computations and Experimental Binding Studies.* The

Journal of Physical Chemistry B, 2013. **117**(51): p. 16443-16454.

144. Jutila, M.A., et al., *L-Selectin Serves as an E-Selectin Ligand on Cultured Human T Lymphoblasts.* The Journal of Immunology, 2002. **169**(4): p. 1768-1773.

145. Ugorski, M. and A. Laskowska, *Sialyl Lewis(a): a tumor-associated carbohydrate antigen involved in adhesion and metastatic potential of cancer cells.* Acta Biochim Pol, 2002. **49**(2): p. 303-11.

146. Strell, C. and F. Entschladen, *Extravasation of leukocytes in comparison to tumor cells.* Cell Communication and Signaling : CCS, 2008. **6**: p. 10-10.

147. Subramaniam, D.R., D.J. Gee, and M.R. King, *Deformable cell-cell and cell-substrate interactions in semi-infinite domain.* Journal of Bomechanics, 2013. **46**(6): p. 1067-1074.

148. Kang, S.-A., et al., *Blocking the Adhesion Cascade at the Premetastatic Niche for Prevention of Breast Cancer Metastasis.* Molecular Therapy. **23**(6): p. 1044-1054.

149. Golubovskaya, V.M., *Focal Adhesion Kinase as a Cancer Therapy Target.* Anti-Cancer Agents in Medicinal Chemistry, 2010. **10**(10): p. 735-741.

150. PARK, J.S., et al., *Resveratrol Inhibits Tumor Cell Adhesion to Endothelial Cells by Blocking ICAM-1 Expression.* Anticancer Research, 2009. **29**(1): p. 355-362.

151. Yin, X., et al., *Knockdown of fucosyltransferase III disrupts the adhesion of circulating cancer cells to E-selectin without affecting hematopoietic cell*

adhesion. Carbohydrate research, 2010. **345**(16): p. 2334-2342.