VERTEX MODELING OF CONFLUENT CELLULAR NETWORKS WITHIN A MOLECULAR DYNAMICS FRAMEWORK IN LAMMPS

by

Ahyo Chang Falick

B.S. University of Colorado Boulder, 2022

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Master of Science

Department of Applied Mathematics

2023

1 Abstract

Falick, Ahyo Chang (M.S., Applied Mathematics)

Vertex Modeling of Confluent Cellular Networks Within a Molecular Dynamics Framework in LAMMPS

Thesis directed by Professor Franck Vernerey

Cell migration is a fascinating biological phenomenon that plays a central role in vital bio-processes for organisms including, but not limited to, immune response, wound repair, and tissue homeostasis. Models of cell migration are also used to emulate harmful epithelial cancers and collective bacterial movement. Many computational models have been proposed about this topic, but they are often very complex to implement and therefore have cost and size limitations. The purpose of this project is to implement an active vertex model in a particle dynamics framework software, known as the Large-Scale Atomic/Molecular Massively Parallel Simulator (LAMMPS). By applying the forces to cell centers instead of cell vertices, we can use particles to simulate cells and use a Dual Voronoi Tessellation to calculate the forces needed. Using LAMMPS allows us to take advantage of its highly parallel nature, providing a framework for larger and more complex systems. Our results include time-step convergence analysis, unjamming transition simulations, and modelled motion of an endothelial cell monolayer on a grooved substrate. Simulating these migratory patterns and understanding their biophysical significance has applications in a wide range of diverse domains including organoid development, stem cell research, and endovascular device design.

Contents

List of Figures

2 Introduction

Collective cell migration (Sadati et al., 2013; Rozman et al., 2020; Friedl and Gilmour, 2009) is a complex phenomena that involves cells closely adhered to each other moving in a highly coordinated manner. Understanding this motion is crucial for a number of biomedical applications regarding tumorogenesis, embryo developments, wound healing and organoid engineering. Despite the existence of many experimental (in-vitro) procedures that aid in deciphering the underlying biophysics of cell tissues, computational models (Wyczalkowski et al., 2012) specifically tailored to study such systems are equally important. Moreover, for systems of cell aggregates, discrete approaches that allow for cell individuality are highly relevant due to the problem's inherent stochastic nature. As opposed to continuum models that look at the global or macroscopic level features, these discrete models are more adept at capturing inter cellular forces. These models are broadly classified into off-lattice and on-lattice models (Nava-Sedeño et al., 2020). Refer to (Fletcher et al., 2017; Van Liedekerke et al., 2015) for more details on different discrete models that have been developed over past decades.

One model that has been tremendously successful in representing confluent biological tissues, such as epithelial and endothelial, is the vertex model (VM) (Fletcher et al., 2013; Alt et al., 2017; Farhadifar et al., 2007; Staple et al., 2010). The success of vertex models lie in their abilities to accurately capture cell-shape and size dependent behaviors from a biophysical perspective. These models allow researchers to closely study the various problems arising in confluent cell networks such as rigidity transition in epithelial tissues (Bi et al., 2015), proliferation dynamics (Lin et al., 2017), rosette formations (Yan and Bi, 2019) and many more.

VMs depict individual cells as polygons defined by a mesh made of vertices and edges. In our case, we apply a vertex model to accurately and efficiently map confluent cell migration and tissue mechanics, with eventual extensions into other important cell functions including growth, death, rearrangement, and division. To achieve such a model, we emulate the previous work of a well-formulated active vertex model (Barton et al., 2017) and integrate their derived forces into a molecular dynamics framework. Molecular dynamics (MD) is a method specific to simulating the motion of atoms and molecules. This method demonstrates the dynamic evolution of a given set of particles by solving their governing equations of motion using numerical methods. Some aspects of cellular behavior have been previously modelled within these particle based frameworks (Milde et al., 2014) but their use has remained limited, especially with regards to confluent cell aggregates.

The main goal of this project is to create and analyze a hybrid vertex model approach for implementation within a particle dynamics framework. The Voronoi tessellation allows us to bridge this gap between cell mechanics and molecular dynamics by constructing a polygonal mesh based on the geometry of cell centers. By treating the particles in an MD system as seed points/cell centers for Voronoi tessellation, we then apply forces to the cell centers and analyze the evolution of the polygons as the system progresses.

We have chosen to implement active vertex model into a Large-Scale Atomic/-Molecular Massively Parallel Simulator (LAMMPS) (Thompson et al., 2022), a popular molecular dynamics simulator, heralding many advantages over classic vertex model packages. These advantages include increased computational efficiency, potential extension into three dimensional systems, accessibility and others discussed later in this paper.

In this thesis, we begin by discussing the background necessary for understanding confluent cell modeling, including the derivation of our proposed vertex model and the addition of cell alignment to introduce active behavior into our systems. We next outline the methods and algorithms used to implement such a model into the LAMMPS source code and discuss the commands needed for compilation. Finally, we present the various results procured during the course of this project, namely, timestep convergence, rigidity transitions, and modelling the confluent cell migration of an endothelial monolayer along a grooved substrate.

While this project has produced stand-alone numerical results, its main function will be to serve as the framework for future research and larger projects. It is our hope that the work done in this project will be improved upon to create more complex and large-scale simulations, capable of accurately and efficiently modeling cellular functionality for a variety of systems.

2.1 Nomenclature

Here we will define the numerous variables necessary for implementing an active vertex model.

2.2 Variables

1. E_{VM}: energy from the vertex model for a given cell in the system
2. N: the number of cells in the system
3. K: the elasticity coefficient. Measures the difficulty to change cell area
4. A: the area of a cell
5. A^0: the preferred area of a cell
6. Γ: the contractility coefficient or perimeter modulus. Measures the difficulty to change cell perimeter
7. P: the perimeter of a cell
8. μ or ν: vertices as given from the Dual Voronoi Tessellation
9. Λ: the junction tension of the edge connecting two vertices
10. l: the length of the edge between two vertices
11. $\boldsymbol{F}$: the force applied to the center of a cell derived from the vertex model energy
12. Ω: the set of vertices for a cell

13. $\boldsymbol{r}$: the position in x,y coordinates of either a vertex or cell center. If two subscripts are given, this is a distance vector between two quantities

14. $\boldsymbol{N}$: the perpendicular unit vector ($\hat{z}$ in this context).

15. $\left[\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{r}}\right]$: the Jacobian matrix

16. $\hat{\boldsymbol{r}}$: a normalized position or distance vector

17. p_0: the shape-index

18. $\bar{\Gamma}$: the normalized contractility coefficient

19. $\bar{\Lambda}$: the normalized junction tension

20. $\left[\frac{dr}{dt}\right]$: the change in position of a quantity with respect to time

21. γ_T: the translational friction coefficient

22. v_0: active velocity

23. $\boldsymbol{n}$: polarity vector of a cell

24. f_a: active force strength

25. γ_R: the rotational friction coefficient

26. E_{Align}: the energy of alignment

27. E_S: energy of alignment from neighboring cells

28. E_G: energy of alignment from a grooved substrate

29. J_S: modulus of alignment from neighbors

30. J_G: modulus of alignment from grooved substrate

31. θ: the angle between a cell and the positive x-direction

32. δt: the timestep

33. τ_S: the torque resulting from minimizing energy of alignment from neighboring cells

34. τ_G: the torque resulting from minimizing energy of alignment from grooved substrate

35. ϵ: the error

36. x: the position of a cell center at the final time of a simulation

37. $E_{analytical}$: the analytical net energy of a system

38. P_R: the preferred area of a cell

39. C: the coordination between two cells

40. O: the orientation of a cell with respect to the positive x axis

3 Vertex model

3.1 Dual Voronoi

To implement a vertex model into LAMMPS, we require a one-to-one correspondence between a given set of cell-centers (seed points) and the polygons representing their 2D cell geometry. To do this, we have chosen to use a Dual Voronoi to partition our lattice into a confluent cell monolayer. This method for cell approximation was first proposed by (Honda, 1978) and has since been widely used as a method for creating vertex meshes. The Dual Voronoi consists of two interrelated concepts, the Voronoi tessellation and Delaunay triangulation.

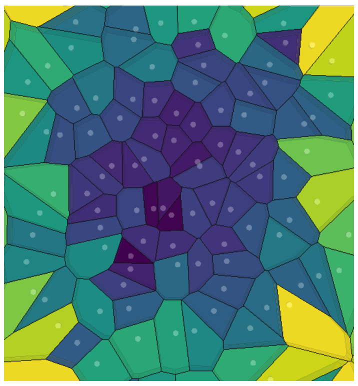
3.1.1 Voronoi tessellation

Given a set of seed points in a plane, a Voronoi tessellation refers to the partitioning of the plane into regions such that any point within the area of each region is closest to the seed point it surrounds. The formal definition is given by

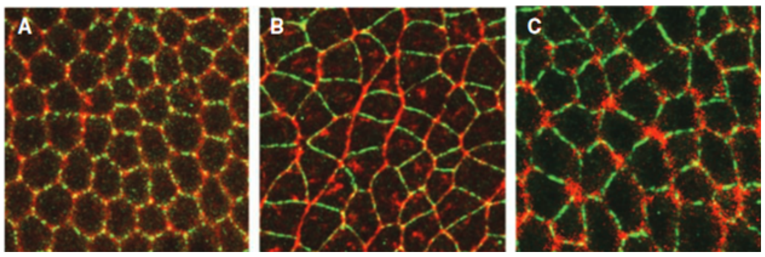
$$R_k = \{x \in X \mid d(x, P_k) \leq d(x, P_j) \ \forall \ j \neq k\}$$

Here, R_k is the region made up of points x corresponding to the cell surrounding seed point P_k. X is the set of all points in the lattice and P is the set of given seed points. The function $d(x, P_i)$ is the Euclidean distance between the coordinates of point x and of seed point P_i.

An illustration of this can be seen in figure 1 below along with an example of a *Drosophilia* cell monolayer undergoing convergent extension.



(a) Voronoi tesselation as visualized in OVITO



(b) Convergent Extension of cells in Drosophilia fly

Figure 1: Comparison between Voronoi tessellation and cellular network

The Voronoi tessellation is very useful for modeling confluent geometries, as it results in a system without gaps between cells. Additionally, each edge between cells is given by a straight line connecting two shared vertices defined by some length and tension. These properties make Voronoi tessellations especially adept at handling rigidity transitions between solid and fluid states. A further analysis of rigidity transitions is discussed in section 3.3.2. As seen in fig. 1, the vertex mesh constructed from the Voronoi tiles closely matches many confluent biological cell networks. The underlying assumption of vertex models is that cell areas can be approximated as polygonal tiles, the vertices of which are subject to motion-inducing forces. In our case, a net force is applied to the cell centers (seed points) themselves and Voronoi tessellations are constructed at each iteration based on the updated particle positions.

3.1.2 Delaunay triangulation

The Dual Voronoi Tessellation also consists of a correspondence known as the Delaunay Triangulation. This is a form of triangular partitioning found by connecting cell centers in such a way that the circumcircle through each triangle contains no other cell centers. Fig. 2 illustrates the Delaunay triangulation for the same system presented in fig. 1a as well as some examples of corresponding circumcircles.

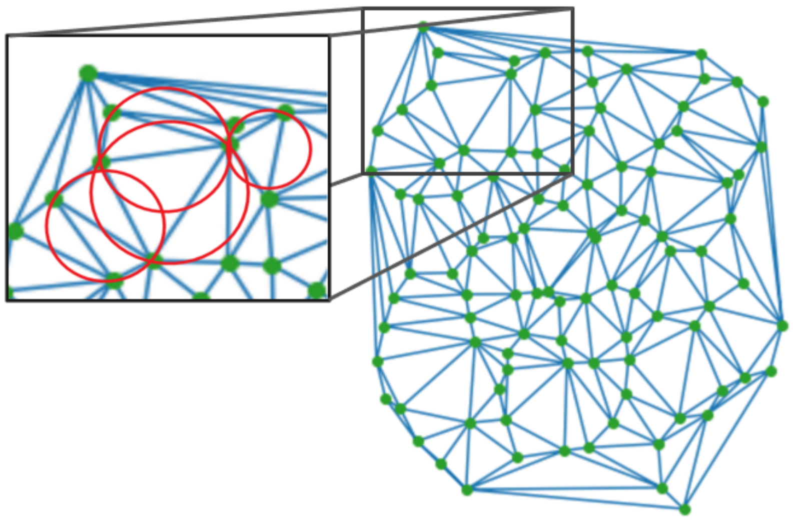


Figure 2: Delaunay Triangulation for same system (Python)

Voronoi tessellation and Delaunay triangulation are considered a duality because the positions of vertices of the Voronoi tessellation can be directly calculated from the valid Delaunay triangles. This results in one vertex shared by three neighboring cells (with the exception of multifold vertices, discussed in section 3.3.1). Each vertex is directly found at the center of each Delaunay circumcircle.

3.2 Kinematics of vertex model

By creating a mesh vertex using the dual Voronoi, we can define certain properties associated with each polygon. The equation that governs the energy of every cell in our vertex model is given by

$$E_{VM} = \sum_{i=1}^{N} \frac{K_i}{2}(A_i - A_i^0)^2 + \sum_{i=1}^{N} \frac{\Gamma_i}{2} P_i^2 + 2 \sum_{\langle \mu,\nu \rangle} \Lambda_{\mu\nu} l_{\mu\nu} \tag{1}$$

For visualization of some of the variables mentioned in this equation, we present a small cellular network made up of polygons, vertices, and edges (fig. 3).

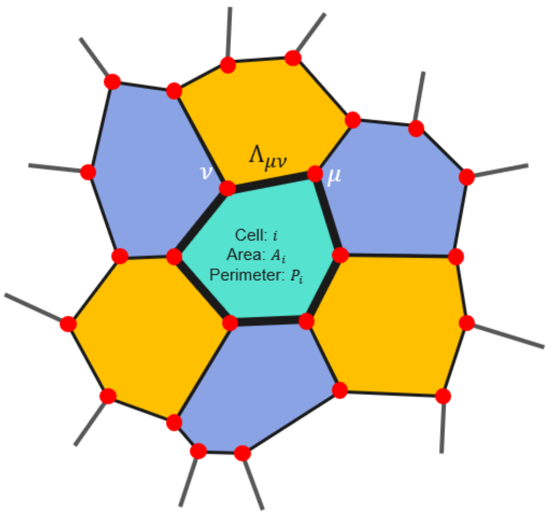


Figure 3: Diagram for understanding vertex model energy

In equation (1), N is the total number of cells and $\langle \mu, \nu \rangle$ is the set of all connected vertices surrounding cell i. Examining the three terms in this expression separately, we have an area, a perimeter, and a junction energy. For the area term, K_i is the elasticity modulus of cell i which measure how difficult it is to change the area of the cell, A_i is the area of cell i and A_i^0 is its preferred area. The perimeter term consists of a perimeter modulus Γ_i measuring how difficult it is to change the perimeter and P_i, the perimeter of cell i. Finally, in the junction term, $\Lambda_{\mu\nu}$ is the tension of the junction between vertices μ and ν and $l_{\mu\nu}$ is the length of that junction.

Each cell junction can be described further as having contributions from two competing mechanisms. The first mechanism comes from the junctional actomyosin network which works to shrink the junction lengths. The second mechanism is driven by cell-cell adhesion which attempts to expand the junction length and area. Depending on which of these processes dominates, $\Lambda_{\mu\nu}$ can be either positive or negative. When $\Lambda_{\mu\nu}$ is positive, cells seek to minimize their overall energy by decreasing the length of the edge, $l_{\mu\nu}$, which signifies contractility is driving junction mechanics. On the other hand, if $\Lambda_{\mu\nu}$ is negative, adhesion is dominant and cells will be inclined to increase their edge lengths to achieve lower energy.

Ideally, individual cells will exhibit different properties within the network, including bulk elasticity K and preferred area A_i^0. Notably, cells undergoing mitosis are modeled by changes in their preferred areas. As is customary in many vertex models, for the purpose of studies in this work we do not consider any cell proliferation and hence, cell number N, along with K, $A_i^0 = A_0$, Γ and Λ are held constant throughout the every simulation.

3.3 Dynamic vertex model

Conventional vertex models are mostly quasi-static and are typically based on energy minimization criteria in which the vertices of the Voronoi tessellation are updated such that the tissue always remains in some minimum energy state (global or local minima). This causes the system to evolve through a sequence of equilibrium configurations at each iteration resulting in a net force on cells always equal to zero. While this approach is suitable for studying some problems, biological tissues most often exhibit out of equilibrium behavior which is why it is important to be able to model tissues when the net force on cells is non-zero. The net force discussed here is derived from minimizing the vertex model energy i.e. by taking the gradient of E_{VM} with respect to cell i,

$$\boldsymbol{F}_i = \nabla_i E_{VM} \tag{2}$$

Doing so, results in applying the following force onto the center of cell i, as derived by Barton et al., (Barton et al., 2017).

$$\mathbf{F}_i = -\sum_{k=1}^{N} \frac{K_k}{2}(A_k - A_k^0) \sum_{v\in\Omega_k} [\mathbf{r}_{v+1,v-1} \times \mathbf{N}_k]^T \left[\frac{\partial \mathbf{r}_v}{\partial \mathbf{r}_i}\right] \tag{3}$$

$$-\sum_{k=1}^{N} \Gamma_k P_k \sum_{v\in\Omega_k} (\hat{\mathbf{r}}_{v,v-1} - \hat{\mathbf{r}}_{v+1,v})^T \left[\frac{\partial \mathbf{r}_v}{\partial \mathbf{r}_i}\right] \tag{4}$$

$$-\sum_{k=1}^{N} \sum_{v\in\Omega_k} [\mathbf{\Lambda}_{v-1,v}\hat{\mathbf{r}}_{v,v-1} - \mathbf{\Lambda}_{v,v+1}\hat{\mathbf{r}}_{v+1,v}]^T \left[\frac{\partial \mathbf{r}_v}{\partial \mathbf{r}_i}\right] \tag{5}$$

This force equation requires definitions for the following newly introduced variables. Ω_k is the set of all vertices belonging to cell k, and ν is a vertex in this set. $\boldsymbol{r}_{\mu,\nu}$ is the vector connecting vertices μ and ν and $\hat{\boldsymbol{r}}_{\mu,\nu}$ is a normalization of the same vector. $\boldsymbol{N}_k$ is the unit vector perpendicular to the area of cell k and because our cells are all 2 dimensional, this corresponds to the unit vector in the z-direction $[0, 0, 1]$. The final term in each sum is the 3×3 Jacobian matrix which describes the transformation between coordinates of cell centers and position of the vertices as per the dual Voronoi.

It should be noted that the multiplication of each term by the Jacobi, $\left[\frac{\partial \mathbf{r}_v}{\partial \mathbf{r}_i}\right]$, ensures that the forces acting from cell k onto cell i will be zero unless k and i are direct neighbors of one another. Furthermore, the inner sum terms $\sum_{\nu\in\Omega_k}$, will be zero for all vertices not shared by both cell i and cell k.

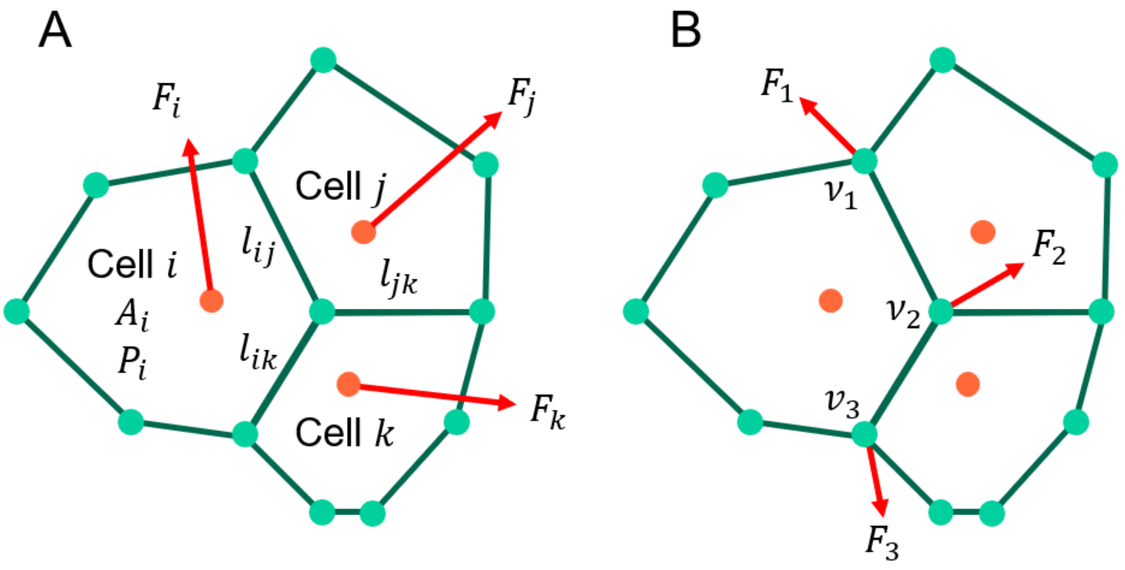


Figure 4: (A) shows a VM where the forces are applied to cell centers and (B) shows a VM with forces applied to vertices

Unlike typical vertex models, we apply the force in equations (3-5) directly on the cell centers. Fig. 4 shows the difference in these methods of force application. This distinction is important to our framework, because it allows us to apply our VM for simulation within a molecular dynamics software where forces are applied to particles.

3.3.1 Rigidity transitions

One of the most impressive features of vertex models is their ability to model epithelial tissues in both solid and fluid states. This behavior is not just a mathematical outcome of vertex models, but rather is found to exist in the tissues of many biological organisms. For example, during embryo development, tissues transition from a fluid-like state to a more solid-like state. Furthermore, this transition behavior is not particular to living biological tissues but is also found in a variety of passive materials such as colloids, foams and granular materials. In these passive materials, the solid state is often referred to as the jammed state because particles are unable to move freely due to crowding. On the other hand, when materials are allowed to flow via high rates of neighbor exchange, they are said to be in a fluid or unjammed state. Since the transition between jammed and unjammed states is associated with changes in rigidity of the aggregate, they have been termed *rigidity transitions*. Unlike rigidity transitions in passive materials that are driven by density, the state of biological tissues is reliant on many other factors.

In vertex models their are various sources of tissue fluidity that drive the rigidity transition. One notable source of state reliance comes from the activity of cell proliferation and apoptosis. It has been well established that cell division and death result in increased tissue fluidization. (Bi et al., 2015) showed that rigidity transitions can also exist in non-proliferating tissues. In such cases, cell rearrangements (T1 transitions) act as fluidizing agents. A crucial parameter that sets the frequency of these neighbor exchanges (and hence differentiates systems in the fluid state from those in the solid state) is the dimensionless shape-index, given by p_0, defined by

$$p_0 = -\frac{\bar{\Lambda}}{2\bar{\Gamma}} \tag{6}$$

where

$$\bar{\Gamma} = \frac{\Gamma}{K(A^0)} \quad \text{and} \quad \bar{\Lambda} = \frac{\Lambda}{K(A^0)^{3/2}} \tag{7}$$

are normalised contractility and normalised tension respectively. In other words the material parameters pertaining to actomyosin contractility (junctional as well as cortical) and strength of cell-cell adhesion dictates how frequently cells rearrange, consequently setting the phase in which the tissue exists. In vertex models the critical value at which rigidity transitions occur is found to be $p_{crit} \approx 3.772$. When p_0 is below this value, the tissue exhibits cells behaving in a solid state and when p_0 is above p_{crit} cells behave more like a fluid. From a mechanical perspective the solid state can be attributed to possess finite elasticity and shear modulus which signify that work will be required to induce deformation. On the other hand, fluid-state tissues are attributed by negligible shear resistance which is why they are also referred to as soft-network. Fig. 5 provides visualization of these regimes as a function of normalized contractility and tension.

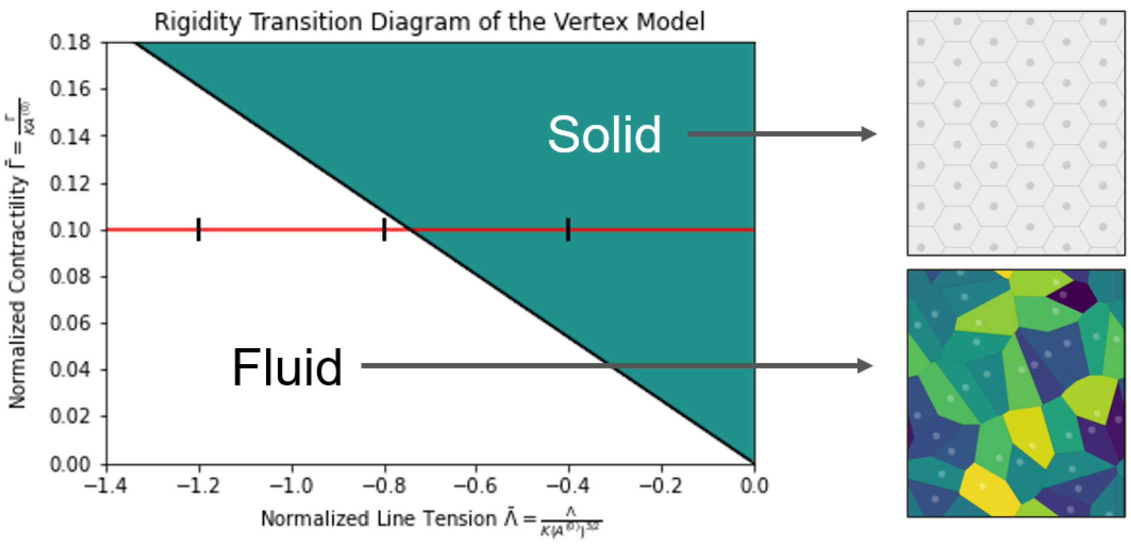


Figure 5: Rigidity diagram as a function of $\bar{\Lambda}$ and $\bar{\Gamma}$

The black line through the middle of this diagram represents the critical shape index at which jamming and unjamming transitions occur. To get a better idea of the differences between systems in the solid and fluid states, we present 4 sample systems at different points along the horizontal red line (fig. 6).

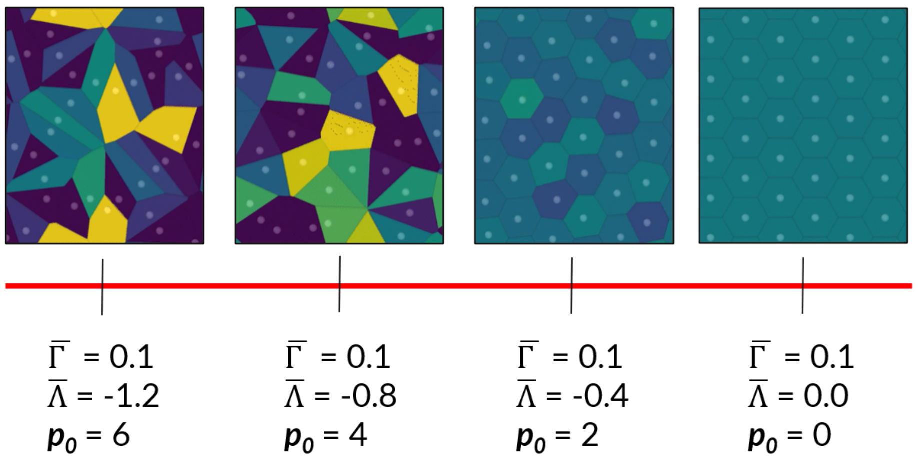


Figure 6: Simulations at four different values of $\bar{\Lambda}$

Here, $\bar{\Gamma}$ is held constant at 0.1 and $\bar{\Lambda}$ is varied. The first two systems exhibit fluid state behavior while the last two exist in the solid state. Each of the cells in these plots is colored based on cell area.

3.3.2 T1 transitions and multifold vertices

For complete dynamic vertex modeling, it is necessary for cellular networks to exhibit neighbor exchanges. These neighbor exchanges, called T1 transitions, occur when cells undergo rearrangement. These exchanges are a crucial part of endothelial cellular network modeling. During this neighbor exchange, the integrity of the monolayer is maintained while the tissue undergoes a deformation. Well-studied occurrences of this remodeling includes convergent extension in *Drosophilia* ((Heisenberg and Bellaïche, 2013)). A simple schematic is presented in fig. 7 for better understanding of this process.

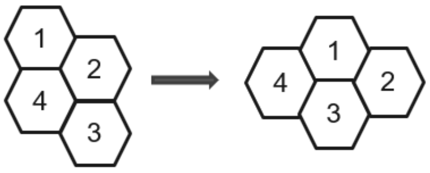


Figure 7: Simple schematic of neighbors exchanging during T1 transition. Cells 1 and 3 are originally neighbors, but as the network evolves, 2 and 4 become neighbors

Neighbor exchanges occur regularly throughout cellular networks in both fluid and solid states, so handling T1 transitions is an integral part of modeling tissue mechanics and morphogenesis. By constructing our vertices around the dual Voronoi tessellation at every iteration and accounting for cases with multifold vertices, we ensure system restructuring that allows for any number of these transitions to occur at any time.

3.4 Cell alignment

For a more complete model of confluent cell migration, it is necessary for our systems to exhibit active behavior. We can more realistically model active cellular networks by introducing cell alignment given by the polarity of each cell. A central model for describing active matter in systems that exhibit collective motion and swarming, is the Vicsek model where self-propelled particles locally align with their neighbors. This model is used to describe observed behaviors for a wide range of organisms and animals, including herds of mammals, flocks of birds, colonies of fire ants, and configurations of bacteria colonies (Chaté et al., 2008). In fire ants, rafts are constructed using ant bodies and each of their preferred directions are predicted using the Vicsek model (Wagner and Vernerey, 2022).

3.4.1 Background and motivation

In this context, we emulate active behavior within tissues by applying the Vicsek model to introduce a polarity term for each cell i given by normalized vector $\boldsymbol{n}_i$. This vector determines the direction of motion for every cell and can depend on a variety of alignment models. For our purposes, we treat this term as both a social and an environmental force. Of the many environmental forces and three main social forces (collision-avoidant, long-range and neighboring) we seek to replicate the effects of neighboring cell motion for a monolayer on a grooved substrate.

By introducing terms of alignment, we extend our VM to exhibit similar results to experimental observations. In particular, we modeled confluent cell migration patterns on a grooved substrate which follows the experimental observations of (Leclech et al., 2022). Leclech and others observe the motion of a monolayer of unconfined endothelial cells. These systems were found to exhibit properties in the fluid state as well as movement along anti-parallel cell streams without any sustained polarization.

3.4.2 Kinematics

Within our VM, we now introduce an additional polarity term of alignment into the Brownian dynamics equation. Along with our vertex model force, the change in position at every time step is now given by

$$\left[\frac{d\boldsymbol{r}_i}{dt}\right]_n = \frac{1}{\gamma_T}[\boldsymbol{F}_i]_n + [v_0\boldsymbol{n}_i]_n \tag{8}$$

The term $[v_0\boldsymbol{n}_i]_n$ is the polarity force (Asnacios and Hamant, 2012) that is derived from an additional energy minimization term for the new alignment energy given by

E_{Align}. Here, the active velocity is defined as

$$v_0 = f_a/\gamma_R \tag{9}$$

where γ_R is the rotational friction and f_a is the active force strength which measures the cell's ability to be propelled by its own mechanics. High value of f_a would signify that cells would quickly align their direction of motion along the direction that minimizes their alignment energy as discussed in next section.

The new equation of motion (equation (8)) still involves no random noise and thus can be treated and solved as a deterministic ODE. If random motion were to be introduced, the differential equation would become stochastic, requiring a numerical solution derived from the Euler-Maruyama method. In our case, LAMMPS again uses a Forward Euler timestepping scheme.

3.4.3 Alignment energy

We can break down our alignment energy E_{align} into two separate energies. The first, E_S, is the alignment of neighbors and the second, E_G, is defined by the alignment along a groove. These two terms commute resulting in

$$E_{Align} = E_S + E_G \tag{10}$$

The alignment energy from the neighbors of cell i is given by

$$E_S = -J_S \sum_{j=1}^{N_S} \boldsymbol{n}_i \cdot \boldsymbol{n}_j \tag{11}$$

where S denotes that we are working with the energy from surrounding cells and N_S is the number of j cells that are neighbors with cell i. J_S is the neighbors alignment modulus and dictates to what degree the alignment from neighboring cells should affect cell i. The vectors $\boldsymbol{n}_i = [\cos\theta_i, \sin\theta_i]$ and $\boldsymbol{n}_j = [\cos\theta_j, \sin\theta_j]$ are the polarity vectors of cells i and j respectively.

The alignment energy for cell i from the groove is defined as

$$E_G = \frac{J_G}{2}(\sin\theta_i)^2 \tag{12}$$

where J_G is the groove alignment modulus that sets the magnitude of how strongly the cellular motion is influenced by the grooved substrate and θ_i is the angle of polarity vector for cell i measured against the positive x direction in this case. This equation results from a desire for the groove direction to be the direction of minimized energy. Using $\sin^2$ thus ensures that E_G will be at a minimum (0) when the cell's polarity is aligned along either the positive or negative x-axis.

3.4.4 Polarity term

As previously stated, the polarity $\boldsymbol{n}$ of each cell is necessary for determining the new updated position of a cell. This term can be defined in terms of the angle θ between the cell's direction of velocity and the positive x direction. This definition of polarity is given by

$$[\boldsymbol{n}_i]_k = (\cos[\theta_i]_k, \sin[\theta_i]_k, 0) \tag{13}$$

In order to make use of this term, we need to be able to update it after every iteration so that it changes with the system. Doing so involves first updating θ such that the polarity vector, $\boldsymbol{n}_i$, aligns towards the direction of either $[1, 0, 0]$ or $[-1, 0, 0]$. We use a Forward Euler timestepping scheme given by

$$[\theta_i]_k = [\theta_i]_{k-1} + \delta t \left[\frac{d\theta_i}{dt}\right]_{k-1} \tag{14}$$

to achieve the new angle. In this equation, k is the current iteration and δt is the time-step.

To find the change in angle, we first take the gradient of E_{Align}, to arrive at two torques, one for each of our two alignment energies, E_S and E_G. These torques follow the same form:

$$\boldsymbol{\tau}_S = -\boldsymbol{n}_i \times \nabla_{\boldsymbol{n}_i} E_S \tag{15}$$

and

$$\boldsymbol{\tau}_G = -\boldsymbol{n}_i \times \nabla_{\boldsymbol{n}_i} E_G \tag{16}$$

These gradients, $\nabla_{\boldsymbol{n}_i} E_S$ and $\nabla_{\boldsymbol{n}_i} E_G$, map the alignment energies, E_S and E_G, to the direction of most rapid increase and taking the cross product with $-\boldsymbol{n}_i$ ensures we are minimizing them. Because the two alignment energies commute, these torques will also commute to give the total torque, $\boldsymbol{\tau}_i$, which results from minimizing the total alignment energy. We then use $\boldsymbol{\tau_i}$ to directly calculate the change in angle $\frac{d\theta_i}{dt}$ desired by cell i:

$$\frac{d\theta_i}{dt} = \frac{1}{\gamma_R} \boldsymbol{\tau}_i \cdot \boldsymbol{N}_i$$

Here, γ_R is the rotational friction coefficient and $\boldsymbol{N}_i$ is the unit vector perpendicular to the area of cell i, which in our case is $[0, 0, 1]$ since we are working in 2 dimensions. Finally, once we have the new angle, the polarities of each cell are then directly calculated using the equivalence in equation (13).

3.5 Units in vertex models

One topic of interest with respect to vertex models is their relation to real world systems. This gives way to an important question: what are the units of measurement in VMs? In order for a VM to maintain consistency, it is essential for the system to remain unitless. In other words, in order for the system to remain scale-independent, we need to normalize each of the parameters used such that they have no dependence on real world units. In vertex models the quantities can then be expressed in terms of derived units of length ($\sqrt{A_0}$), energy (KA_0) and time ($\gamma_T/(KA_0)$).

4 Vertex modeling in LAMMPS

Vertex models have been developed within various computational frameworks and packages (SAMoS, chaste) over the years, yet their usage and further development have been impeded by intrinsic complexities. Most available vertex modeling tools require programming knowledge, which poses a challenge for researchers from diverse backgrounds who wish to use VMs and interpret the results by comparing them with their own observations (experimental or theoretical). Consequently, the potential of vertex models remains unexploited, as it has only been utilized by a certain class of researchers. In addition, although existing vertex models can handle a variety of biophysical aspects in confluent tissue layers, such as actomyosin contractility (Montell, 2008) and cell-cell interactions in the form of topological transitions (Staple et al., 2010), extension into complicated problems involving full-fledged three-dimensional cell aggregates with a large number of cells and their interactions with surrounding matrix has yet to be implemented. Fortunately, with advancements in computational power over recent years, increasingly larger and more complex systems are being more regularly studied.

To this end, the aim of the current study is to take the vertex model one step further by implementing it within an open-source particle dynamics software called LAMMPS. This simulator gives us many advantages over other vertex models. Firstly, the massively parallel architecture of LAMMPS allows for simulation of large and complex systems that would be otherwise impossible, or at least frustratingly slow. Second, LAMMPS has impressive three-dimensional capabilities. This will allow for future models built from the framework presented here to scale up to 3 dimensions with relative ease. The third advantage for using LAMMPS is that it is entirely open source, so its functionalities are extremely well documented and even researchers with limited programming background can model complex systems by simply learning how to provide the required the inputs. In the context of cellular networks, developing this vertex model within LAMMPS has allowed us to adapt a large number of features that are found to play a crucial role in tissue mechanics and morphology into our systems. Such features include extracellular matrices (ECM), hydrogel matrices, embedding cell-assemblies and chemo-mechanical attributes such as nutrient diffusion.

4.1 Implementation into LAMMPS

The LAMMPS documentation details a vast number of mutable functions used to simulate a diversity of different systems. Among these functions, the two most important in this project are computes (used to calculate some attribute for a group of particles) and fixes (used to set some attribute of a group particles). To adjust these functions, we write and edit input scripts as text files to feed to a Linux terminal for compilation.

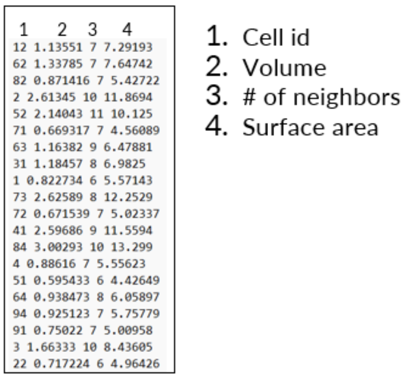
The LAMMPS source code is written entirely in C++. In order to create a working force equation that would be able to be accessed by LAMMPS input scripts, it was necessary to build our own fix command. The following sections detail the various algorithms used to implement this function.

4.1.1 Constructing Voronoi in LAMMPS

LAMMPS does contain its own VORONOI package which has helpful functions for constructing Voronoi tessellations including the function compute voronoi/atom. This compute creates a per-atom array with information for each of the cells in the Voronoi tessellation, including their volume, number of neighbors, and surface area. Additionally, a separate array stores the ids corresponding to each neighbor as well as the junction length between them. Figure 8 shows the function call used in our input script to get this information and the per-atom array that it outputs in a separate dump file.

```
compute     voro all voronoi/atom surface all
compute     v2 all voronoi/atom neighbors yes
```

(a) compute voronoi/atom command as written in the LAMMPS input script



```
 1   2   3   4
12 1.13551 7 7.29193
62 1.33785 7 7.64742
82 0.871416 7 5.42722
2 2.61345 10 11.8694
52 2.14043 11 10.125
71 0.669317 7 4.56089
63 1.16382 9 6.47881
31 1.18457 8 6.9825
1 0.822734 6 5.57143
73 2.62589 8 12.2529
72 0.671539 7 5.02337
41 2.59686 9 11.5594
84 3.00293 10 13.299
4 0.88616 7 5.55623
51 0.595433 6 4.42649
64 0.938473 8 6.05897
94 0.925123 7 5.75779
91 0.75022 7 5.00958
3 1.66333 10 8.43605
22 0.717224 6 4.96426
```

1. Cell id
2. Volume
3. # of neighbors
4. Surface area

(b) Dump file from compute voronoi/atom

Figure 8

Initially, we used the second array containing a full list of neighbors to construct a matrix of Delaunay triangles at each timestep and a corresponding vertex matrix. This method proved to not be robust enough for handing T1 transitions, so it was necessary to develop our own algorithm for finding and storing these vertices. Algorithm 1 shows our approach to this using the properties of the dual Voronoi.

Algorithm 1 Find and Store Vertices

```
for cell i in system do
    vertices = [ ]
    DelaunayTriangles = [ ]
    neighborslist = list of cell neighbors
    for cell j in neighborslist do
        for cell k in neighborslist do
            if j $\neq$ k then
                Form circumcircle from Delaunay Triangulation (DT)
                for cell c in system do
                    if c is found in circumcircle then
                        DT is not valid
                        break
                if DT is valid then
                    $v$ = circumcenter of circle
                    vertex.append($v$)
                    DelaunayTriangles.append(DT)
```

It should be noted that this algorithm, while accurate, is very computationally expensive, with a worst case time complexity of between $O(N^2)$ to $O(N^3)$ where N is the number of cells in the system. For large systems, this is obviously an algorithm that will need to be changed, but for the time being, we felt developing a robust model was more important.

4.1.2 T1 transitions

To handle the T1 transitions discussed in section 3.3.2, it was necessary to make our custom force code totally independent of the LAMMPS compute voronoi output. Constructing our Delaunay triangles and coinciding vertices in this manner allows us to handle these neighbor exchanges. Figure 9 provides a more detailed visualization of the change in Delaunay triangulation that occurs during a T1 transition.

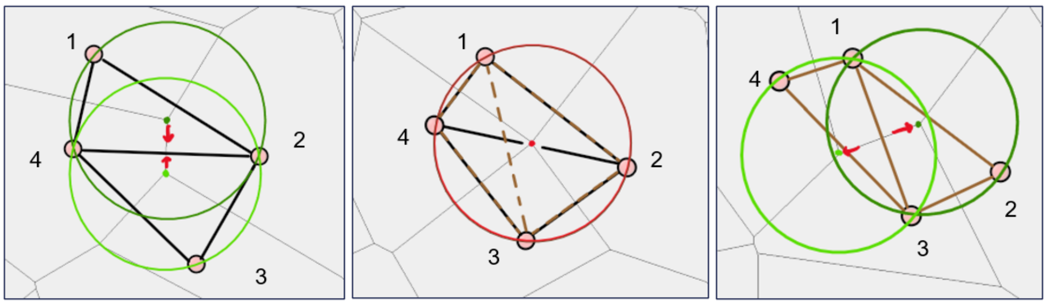


Figure 9: Schematic of changing Delaunay triangles and circumcenters during a T1 transition. First, cells 4 and 2 are neighbors with each other and they both belong to two valid triangles. The vertices given by these triangles collapse into a single vertex in the second frame. Here, there is now 4 valid Delaunay triangles that all correspond to the same cicrumcircle. Finally, in the third frame, the vertex divides into two vertices in such a way that cells 1 and 3 are now neighbors while 2 and 4 are not.

From left to right, this system of four cells evolves in such a way that the edge between cells 2 and 4 converges to 0 forming a multifold vertex shared by all four cells. At this time, compute voronoi/atom no longer recognizes the cell pairings 1 and 3 or 2 and 4 as neighbors which caused problems in our Delaunay triangle matrix construction. By using algorithm 1 instead, we are able to seamlessly transition to and from a multifold vertex configuration for any number of cells.

4.1.3 Ordering vertices

Once we have a list of vertex positions and the corresponding list of Delaunay triangles represented by a triad of cell centers, we use these lists to create a two dimensional vector where each row is a list of the vertices of a cell in the system. As depicted in the force equation, a consistent order for traversing the set of vertices for a cell must be established. Specifically, each of the vertex distance vectors $\boldsymbol{r}_{\mu\nu}$ require such a loop. The below algorithm provides a framework for ordering vertices in a counterclockwise manner using the previously formed cell-vertex vector.

Algorithm 2 Order vertices given unordered vertex list

```
StartVertex = vertex with minimum y coordinate
if multiple vertices have same y coordinate then
    StartVertex = vertex with minimum x and y coord
$\bar{x} = [1, 0, 0]$
MinAngle = $2\pi$
vertexlist.pop(StartVertex)
orderedlist.push(StartVertex)
while vertexlist is not empty do
    for v in vertexlist do
        vec = v - StartVertex
        $\theta = \arccos\left(\frac{vec}{|vec|} \cdot \bar{x}\right)$
        if $\theta <$ MinAngle then
            MinAngle = $\theta$
            NextVertex = v
    vertexlist.pop(NextVertex)
    orderedlist.push(NextVertex)
return orderedlist
```

4.1.4 Polarity term implementation

Here, we discuss our implementation of the polarity term defined in section 3.4.4. To update these polarities at each iteration, we first update the angle θ then find the polarities using equation (13). As LAMMPS does not keep track of this angle for each cell center/particle as the simulation progresses, we have devised our own per-atom array containing the updated θ at each iteration. An important part of our polarity calculations is properly initializing the polarities. If these polarity directions are not randomly distributed from 0 to 2π, this might cause some bias in the direction which cells align to. For example, if there is a bias towards polarity in the first quadrant (0 to $\pi/2$), the cells will overwhelmingly choose to follow the direction of the positive x-axis when aligning with the grooved substrate. This type of alignment results in no visible anti-parallel cell streams.

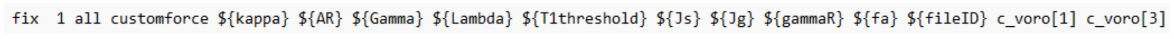
4.1.5 Adding forces using fix customforce

At each iteration of the simulation, our function `fix customforce` creates a Voronoi tessellation given the coordinates of the seed points, determines and stores the positions of vertices for each cell, and calculates the forces acting on each cell center using force equations derived from both vertex model and alignment energies. A pseudocode for this fix is presented below, briefly outlining our method of implementation.

Algorithm 3 Pseudocode for fix custom force

```
1. Construct the Dual Voronoi Tessellation for a given configuration of cell centers
2. Find and store the Delaunay Triangles and associated vertices in separate matrices
3. Create a matrix of vertices associated with cell ids
4. Order the vertices of the cell-vertex matrix
  $K$ = elasticity
  $A^0$ = preferred cell area
  $\Gamma$ = contractility
  $\Lambda$ = tension
  $J_S$ = neighbors alignment modulus
  $J_G$ = groove alignment modulus
  $f_a$ = active force
  $\gamma_R$ = rotational friction coefficient
  for each cell $i$ in the system do
      create an array of neighbors of cell $i$
      $\boldsymbol{F}_i = [0, 0, 0]$
      //* FORCE FROM VERTEX MODEL *//
      for every cell $k$ in the neighbors list do
          $A_k$ = area of cell $k$
          $P_k$ = perimeter of cell $k$
          $innersum1 = [0, 0, 0]$
          $innersum2 = [0, 0, 0]$
          $innersum3 = [0, 0, 0]$
          for vertex $v$ in the vertices of cell $k$ do
              if $v$ also belongs to cell $i$ then
                  $innersum1 \mathrel{+}= [\mathbf{r}_{v+1,v-1} \times \mathbf{N}_k]^T \left[\frac{\partial \mathbf{r}_v}{\partial \mathbf{r}_i}\right]$
                  $innersum2 \mathrel{+}= (\hat{\mathbf{r}}_{v,v-1} - \hat{\mathbf{r}}_{v+1,v})^T \left[\frac{\partial \mathbf{r}_v}{\partial \mathbf{r}_i}\right]$
                  $innersum3 \mathrel{+}= [\mathbf{\Lambda}\hat{\mathbf{r}}_{v,v-1} - \mathbf{\Lambda}\hat{\mathbf{r}}_{v+1,v}]^T \left[\frac{\partial \mathbf{r}_v}{\partial \mathbf{r}_i}\right]$
          $\boldsymbol{F}_i \mathrel{+}= -\frac{K}{2}(A_k - A^0) \times (innersum1) - \Gamma P_k \times (innersum2) - innersum3$
      //* FORCE FROM ALIGNMENT *//
      $\tau_G = -J_G * \sin(\theta_i) * \cos(\theta_i)$
      $\tau_S = 0$
      for every cell $k$ in the neighbors list do
          $\tau_S \mathrel{+}= \cos(\theta_i) * \sin(\theta_k) - \sin(\theta_i) * \cos(\theta_k)$
      $\tau_i = J_S * \tau_S + \tau_G$
      $\left[\frac{d\theta}{dt}\right] = \frac{\tau_i}{\gamma_R}$
      $\theta_i = \theta_i + dt * \left[\frac{d\theta}{dt}\right]$
      $\boldsymbol{F}_i \mathrel{+}= f_a * [\sin(\theta_i), \cos(\theta_i), 0]$
  return $\vec{\boldsymbol{F}}$
```

This algorithm dictates our overall implementation and is called in LAMMPS using the following input script command (fig. 10).



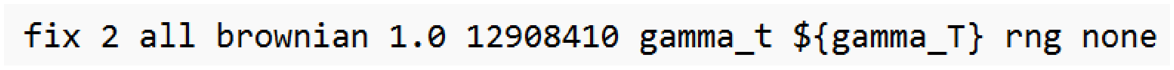
```
fix  1 all customforce ${kappa} ${AR} ${Gamma} ${Lambda} ${T1threshold} ${Js} ${Jg} ${gammaR} ${fa} ${fileID} c_voro[1] c_voro[3]
```

Figure 10: fix customforce command as called in our LAMMPS input script

Here, we feed `fix customforce` each of the parameter inputs defined at the beginning of algorithm 3 as well as information from compute voronoi/atom. Specifically, we give our function the 2nd and 3rd columns of the per-atom array in fig. 8b, which contains the necessary information to find the area and perimeter of each cell. Note that finding the area and perimeter is straightforward using only the vertex coordinates, but to avoid the extra calculation costs, we use compute voronoi/atom to get this information.

4.2 Updating positions and polarities

Finally, we discuss how LAMMPS actually updates particle positions based on the forces found in the previous section. To update cell center positions, we use the fix Brownian function call in LAMMPS show in figure 11.



```
fix 2 all brownian 1.0 12908410 gamma_t ${gamma_T} rng none
```

Figure 11: Fix Brownian as written in the LAMMPS input script

This function updates positions and velocities of every cell center in our system using a Forward Euler timestepping scheme. Forward Euler follows the basic form

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k + \delta t \left[\frac{d\boldsymbol{r}}{dt}\right]_k \tag{17}$$

where $\boldsymbol{r}_k$ is a position vector at the kth iteration using a timestep of δt. The change in particle position $\left[\frac{d\boldsymbol{r}}{dt}\right]$ is found using

$$d\boldsymbol{r} = \gamma_t^{-1}\boldsymbol{F}dt + \sqrt{2k_BT}\gamma_t^{-1/2}d\boldsymbol{W}_t \tag{18}$$

In our case, we do not use any random noise (note the rng none parameter in figure 11) so the term $\sqrt{2k_BT}\gamma_t^{-1/2}d\boldsymbol{W}_t$ is set to zero. This results in a deterministic ODE.

We note that, while Forward Euler comes with potential accuracy issues, implementing a more modern time integrator is not within the scope of this project and is therefore left as an area of potential future work.

5 Results

This section will be devoted to the various numerical results developed throughout this project. The first two sections detail results achieved solely using forces taken from the vertex model (i.e. without polarity or random fluctuation) and the final section gives results after adding a mechanism of alignment to our governing equation of motion. To start, we present a timestep convergence analysis that will yield the range of timestep to be used in all of our simulations. This section is split further into two subsections for separate timestep analysis of both the solid and fluid state. Next, we discuss Rigidity Transitions including the effects of material parameters and take a closer look at rosette formations in the fluid state. Finally, we analyze resulting simulations after alignment is introduced, by looking at coordination and orientation plots for varying levels of neighbor and groove alignment moduli.

5.1 Timestep convergence

For validity in any simulation, it is important to conduct a timestep convergence analysis. This involves measuring the effects of decreasing timesteps by analyzing some quantifiable property of the system. Here, we investigate two different approaches. The first is an energy convergence analysis and the second involves error analysis of cell center positions at the end of each simulation. For energy converge, we plot the previously defined E_{VM} (see equation (1)) with respect to time (in LAMMPS' units) for decreasing timesteps. In the second approach, we present a log plot of the error between the sums of final cell center positions as a function of increasing order of timestep. This method follows work from (Kursawe et al., 2017) which uses the following error equation

$$\epsilon_k^{t_f} = ||\sum_{j=1}^{N} x_j^k - \sum_{j=1}^{N} x_j^{k-1}|| \tag{19}$$

Where $\epsilon_k^{t_f}$ is the error between cell positions at timestep k and previous timestep $k-1$. The superscript t_f denotes that we are looking at the positions of cells at the final time t_f. N is the total number of cells in the system at t_f and x_j is the final position vector of the cell center belonging to cell j. $||\cdot||$ is the 2-norm of a position vector given by $||\bar{x}|| = \sqrt{x_1^2 + x_2^2}$.

5.1.1 Solid state timestep convergence

We will start by applying our two approaches to the solid state, analyzing the system given by the following initial and final cell configuration.

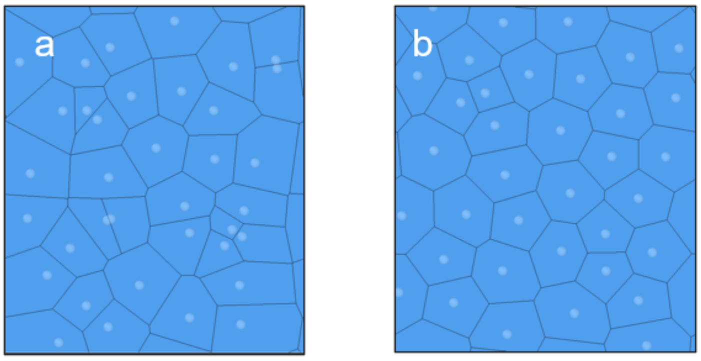
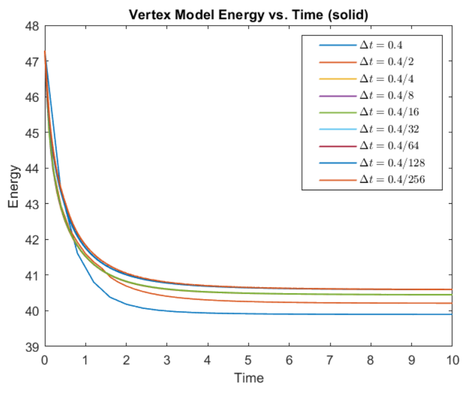
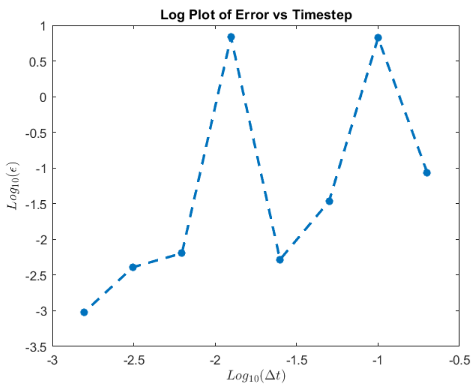


Figure 12: (a) The Voronoi tessellation of the starting system of cell center coordinates. (b) The Voronoi tessellation of the coordinates at the final time. Here we are running for a total time of 10 LAMMPS time units which is unitless, but can be given by the following equivalence: $\tau^* = \tau\sqrt{\frac{\epsilon}{m\sigma^2}}$, where ϵ is the energy, m is mass, σ is distance, and τ is time. We are also using area modulus $K = 1.0$, normalized contractility $\bar{\Gamma} = 0.18$, and normalized tension $\bar{\Lambda} = -0.1$, such that the dimensionless target shape-index is $p_0 = -0.28$ and the system lies in the solid regime

For this solid state timestep analysis, we look at the variability when using 9 different timesteps for simulations spanning a total time of 10. Starting from a timestep of $\Delta t = 0.4$, we decrease Δt by a factor of 2 for each simulation, ending with $\Delta t = \frac{0.4}{2^7} = \frac{0.4}{256}$. Our results from this investigation are given in figure 13.



(a)



(b)

Figure 13: Time step convergence analysis for simulations in the solid state

In fig. 13a, each of the plots exhibit clear asymptotic behavior to around the same analytical energy. The similarities between lines in this plot suggest that decreasing timesteps causes no significant changes between simulations. The error plot for this system is presented in fig. 13b. While this graph plot does not show convergence beyond a reasonable doubt, it is clear that when the timestep lies below an order of 10^{-2} the error is also very small (less than 10^{-2}). From this, we conclude that using a timestep around this size will result in stable simulations.

5.1.2 Fluid state timestep convergence

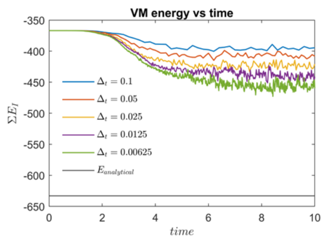
For the fluid state, we again, used simulations of total time 10, but this time with 5 distinct timesteps ranging from $\Delta t = 0.1$ to $\Delta t = 0.1/16$. Figs. 14a and 14b show our energy over time and error vs timestep plots respectively. In the energy over time plot, we add an additional line (grey) indicating the analytical minimum energy for this system found using the equation

$$E_{analytical} = N\left(\frac{\Gamma}{2}P_R^2 + \Lambda P_R\right) \tag{20}$$

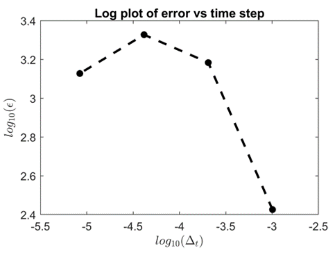
where N is the number of cells in the system and Γ and Λ are the contractility and tension parameters respectively. Here, P_R is the preferred perimeter of the cells given by

$$P_R = -\frac{\Lambda}{2\Gamma} \tag{21}$$

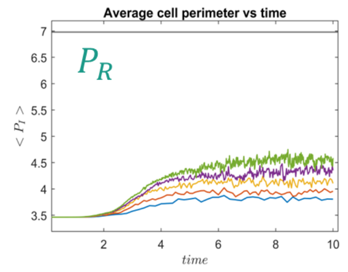
Equations (20) and (21) together imply that the equilibrium configuration in fluid state is give by cells in their preferred areas (i.e. $A_i = A0$) and preferred perimeters ($P_i = P_R$) We add an additional plot (fig. 14c) to show the average perimeters of each cell compared their preferred perimeters (also indicated in grey at around $\langle P_R \rangle = 7$).



(a)



(b)



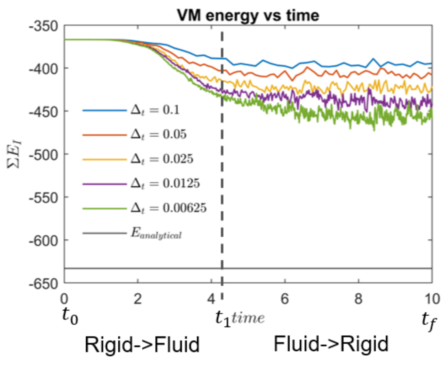
(c)

Figure 14: (a) the vertex model energy given by equation (1) vs. time. 5 simulations at different timesteps are plotted ranging from 0.1 to 0.00625. The gray line towards the bottom shows the analytical solution to the energy equation. (b) a log-log plot of the error between timesteps. (c) Plot of average cell perimeter over time.

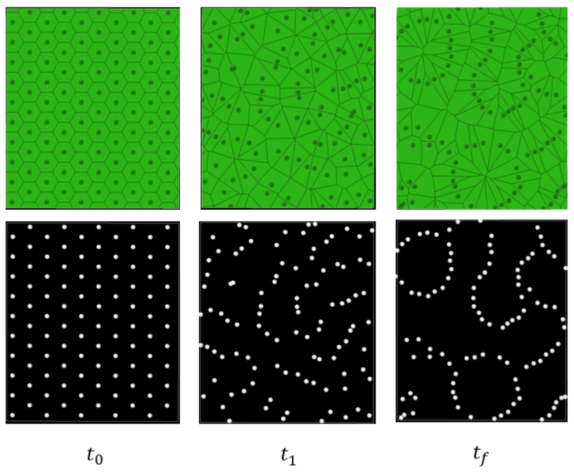
In fig. 14a, the energy clearly does not converge to the analytical energy from equation (20). This energy minimum signifies the global minimum for the system. We theorize that the reason the system does not converge to this minimum is because it is instead converging to a local energy minimum.

Likewise, fig. 14c, shows the average perimeter of cells over time per equation (21). Again, none of the simulations converge to this preferred perimeter, but rather to a local average perimeter that increases as the timestep decreases.

The reason for convergence to a local energy and average perimeter is best visualized by rosette formations throughout the system. For a closer analysis of these structures, we again show the energy over time plot, but now with snapshots of the simulation at three different times (fig. 15). Fig. 15b shows the snapshots of the system along with images of just cell centers for a better idea of how rosettes are formed.



(a)



(b)

Figure 15: Rosette formations

At time $t_0 = 0$ the system is in near equilibrium except for the slight displacement of a single cell center. This displaced cell, along with fluid state parameters, propels the system into a completely fluid configuration as shown at $t_1 \approx 4.2$. As the system evolves further, rosettes form as the system converges to a local energy minimum, shown in the snapshots of the system at $t_f = 10$. In figure 15a, we indicate where each of these snapshots occurred for ease of comparison to the energy of the system.

From this, we can divide the simulation into two regions separated by t_1. Between t_0 and t_1, the system devolves from a near solid state into a fluid one which is reflected by a rapid energy decrease. Between t_1 and t_f, the system converges back to a quasi-solid state and the energy begins to decrease asymptotically towards a local minimum.

5.2 Rigidity transitions

Now that we have chosen a sufficient timestep for both the solid and fluid states, we can examine cell-system behavior in each of these regimes. The solid state can be defined by well-ordered cells with minimal T1 transitions. On the other hand, the fluid state is defined by a much higher rate of neighbor exchange. As previously stated in section 3.3.1, the state that our system falls into is determined by a unitless shape index as defined by $p_0 = -\frac{\bar{\Lambda}}{2\bar{\Gamma}}$.

5.2.1 Solid state

The images below (fig. 16) show the progression of cells in a solid state as they converge to the global energy minimum. Here we use a normalized contractility of $\bar{\Gamma} = -0.1$ and a normalized edge tension of $\bar{\Lambda} = 0.18$ resulting in $p_0 = 0.9$, falling within the solid regime.

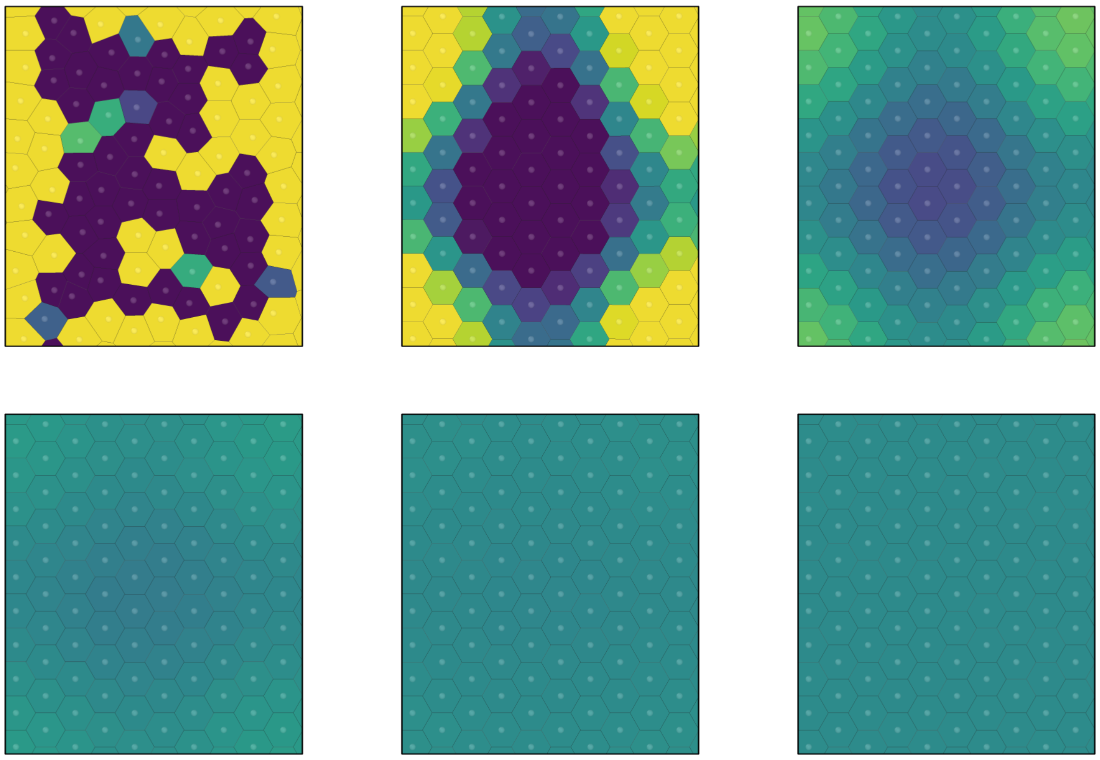


Figure 16: Solid state simulation where cells are colored based on volume

In the simulation above, cell centers start slightly displaced from a perfect hexagonal distribution and cells are colored based on their areas. This is done for visualization of the convergence of this system to a lattice of cells with the same area. Once the cells reach their final configuration, they exhibit no net forces on their cell centers indicating the system has reached a state of minimum energy. While cells in this network all converge to the same area, this is not necessarily the case for all solid state simulations. If the cells are displaced even more at the start, the system will converge instead to a local minimum energy and they will not exhibit identical volumes.

5.2.2 Fluid state

Next, we look at a fluid state simulation starting with a near perfect distribution of cell centers with the exception of one slightly displaced center indicated in red (fig. 17). To achieve the fluid state, we use $\bar{\Gamma} = 0.1$ and $\bar{\Lambda} = -1.5$ resulting in a $p_0 = 7.5$. Note that this is the same simulation given by the green line in fig. 14.

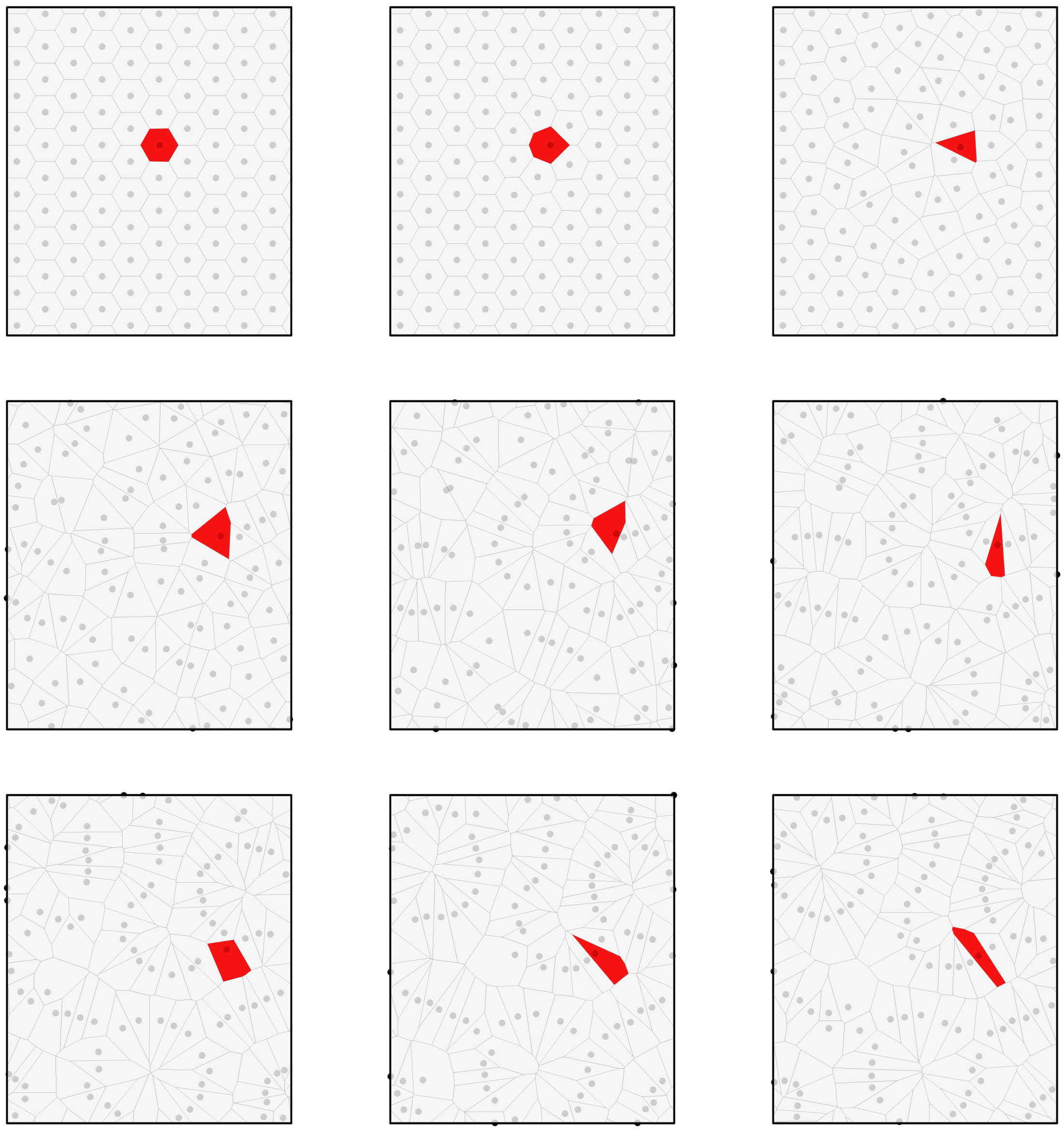


Figure 17: Fluid state simulation

This fluid state simulation is shown to converge to a configuration that is heavily governed by rosette formations. While intuition might suggest that cells in the fluid state should always remain fluid-like and unjammed, recent work from Yan and Bi indicates the formation of these rosette structures show fluid state stabilization ((Yan and Bi, 2019)). They conclude that these rosettes demonstrate a desire for the tissue to transition into the solid-state. This matches the results found from our timestep convergence analysis where we showed that rosette structures form as the system converges to a local energy minimum.

Rosettes are found in a variety of real biological tissues. They occur especially frequently during embryonic development where the tissue rapidly elongates. Notable examples of rosettes found in nature include embryonic elongation of the *Drosophilia* germband (Blankenship et al., 2006), the visceral endoderm of mouse embryos (Trichas et al., 2012), as well as the brains of adult mammals (Zhang et al., 2001). Many studies have shown that these elongations are facilitated by a majority of epithelium participation in rosette structures. In real biological tissues, rosette formations are driven by the contraction of actomyosin networks and propel efficient tissue restructuring and growth (Yan and Bi, 2019).

5.3 Collective cell migration

For numerical results of active cell behavior after alignment is introduced, we develop coordination and orientation plots that show the effects of polarity on a fluid state system. Here, we run simulations at varying levels of J_S and J_G to see how much these variables affect the alignment of our system. Here, we are analyzing the same fluid state system from the previous section where one particle in a 100-cell system is slightly displaced. We run this system for total time of 5 at a timestep of $\Delta t = 0.1/8 = 0.0125$.

To better visualize these results, we have plotted the cell trajectories over time and add a color gradient to the trajectory segment at each time step based on one of two calculations. For neighbor alignment we look at the inter-neighbor cellular coordination and for groove alignment we look at orientation of cells with respect to the x-axis. The following subsections detail the calculations behind each of these plot types as well as our results. For consistency, we hold the fluid state parameters, timestep, and total time constant for each analysis while varying levels of J_S and J_G.

5.3.1 Coordination

First, we look at the coordination of our cells. Specifically, we are looking at the polarity of one cell with respect to the cells neighboring it. The equation we use to calculate coordination between two cells is

$$C_{ij} = \arccos\left(\frac{\boldsymbol{v}_i}{|\boldsymbol{v}_i|} \cdot \frac{\boldsymbol{v}_j}{|\boldsymbol{v}_j|}\right) \tag{22}$$

where C_{ij} is the coordination between cells i and j and $\boldsymbol{v}_i$ is the velocity vector of cell i. We then take the average of this coordination from all the neighboring cells to get the total coordination of cell i.

$$C_i = \langle C_{ij} \rangle = \frac{\sum_{j=1}^{N_S} C_{ij}}{N_S} \tag{23}$$

This value of coordination ranges from -1 to 1 and is used as a color gradient. A coordination of -1 indicates that cell i is moving in the direction opposite to each of its neighbors and a value of 1 suggests it is totally aligned with its neighbors.

5.3.2 Orientation

The orientation of the cells can be calculated in a similar way to the coordination, but now we are looking at the angle between the velocity direction of cell i and the direction of the groove. In our case, we are simulating grooves along the x-axis, so we use

$$O_i = \arccos\left(\frac{\boldsymbol{v}_i}{|\boldsymbol{v}_i|} \cdot \bar{\boldsymbol{x}}\right) \tag{24}$$

where $\bar{\boldsymbol{x}}$ is the unit vector in the positive x direction. As per the definition, the range of O_i is from $0 - \pi$ in radians or $0 - 180$ in degrees. Using $\bar{\boldsymbol{x}}$ allows us to see whether the cells are choosing to align in the positive or negative x direction, which is helpful for visualizing anti parallel cell streams.

5.4 Plots

Now that we have defined coordination and orientation we present various trajectory plots resulting from changing the alignment energy moduli. Each of these figures were produced using Matlab. To create them, we first calculated the coordination and orientation in `fix customforce` and outputted the information to a series of dumpfiles that were then fed to our Matlab code for post processing.

5.4.1 No polarity

First, we show the coordination and orientation plots for a simulation without alignment (fig. 18).

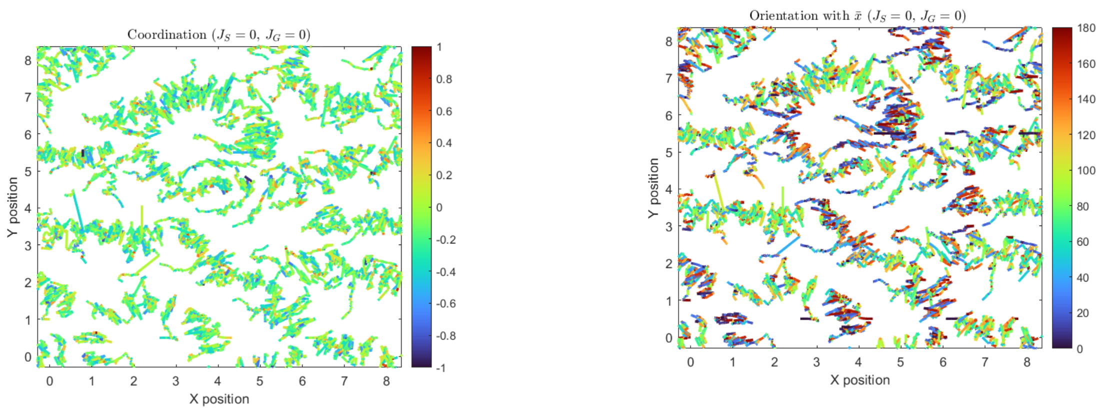


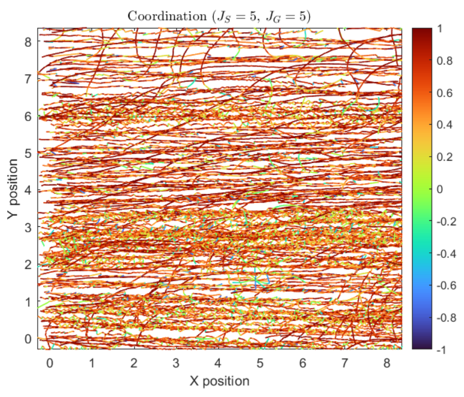
Figure 18: Coordination and orientation plots for fluid state with 0 alignment force

Clearly these plots exhibit no discernible alignment patterns. The trajectories of the cells follow the path of motion described solely by the vertex model energy. In the coordination plot, the trajectories are colored by an overwhelming green, indicating a lack of alignment with any of their neighbors. In the orientation plot, the wide spectrum of color distribution suggests that direction of cells are random with respect to the x axis.

5.4.2 Added alignment

After alignment is added, the cells exhibit totally new behaviors. The plots in fig. 19 show the new trajectories of cells after we introduce alignment using $J_S = J_G = 5$

and an $f_a = 10$.



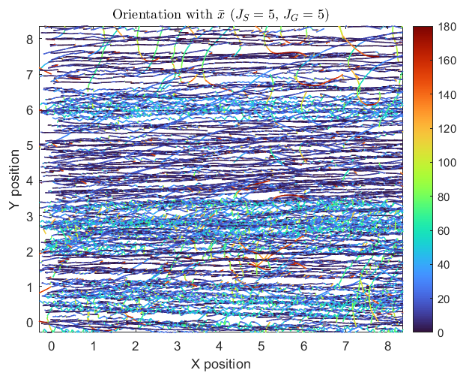


Figure 19: Coordination and orientation plots for fluid state with $J_S = 5$ and $J_G = 5$

These plots depict clear convergence to a singular cell stream. In this case, the system does not exhibit anti-parallel cell streams because of a strong neighbor alignment modulus that propels cells in the direction of alignment with the other cells in the system. The majority of cells initially move in the positive x direction and consequently, the others eventually do the same.

5.4.3 Effects from neighbor alignment

For this section, we look solely at the coordination plots when the system has no groove alignment ($J_G = 0$) and the neighbor alignment modulus is increased from $J_S = 1$ to $J_S = 10$. These four coordination plots are presented below in fig. 20.

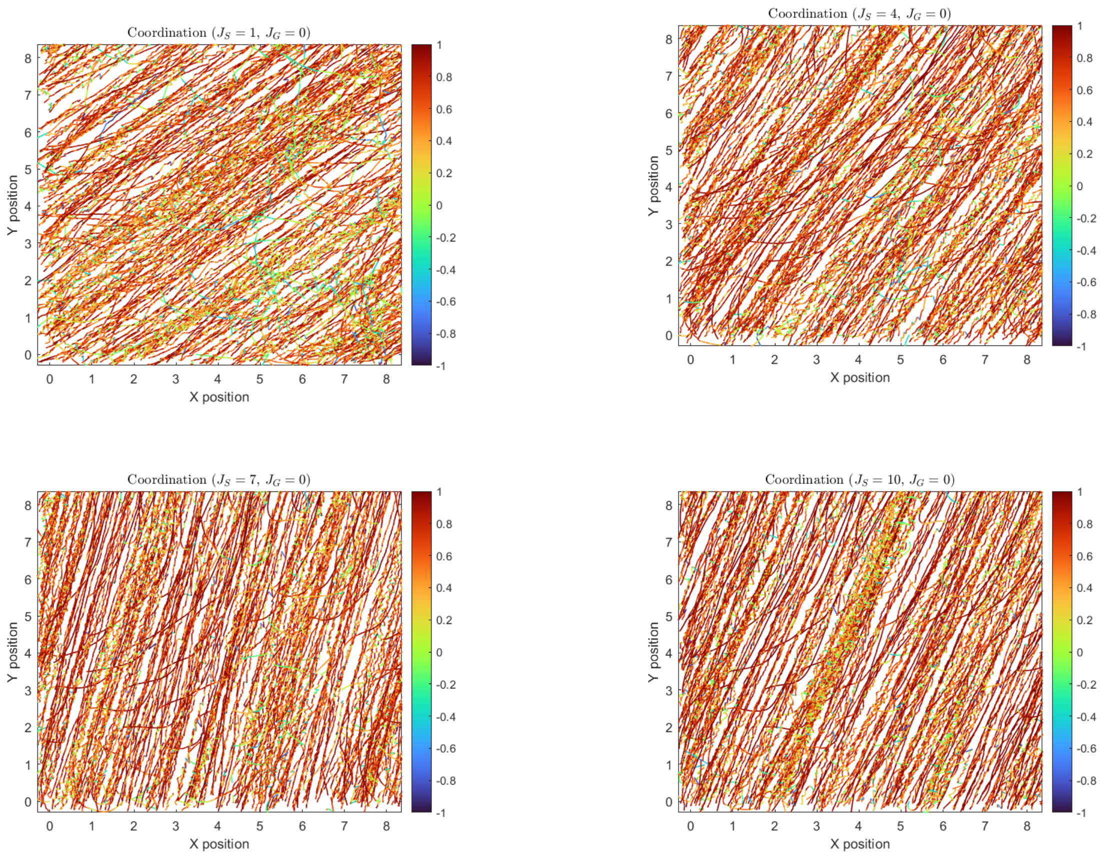


Figure 20: Coordination plots using $J_G = 0$ and $J_S = 1, 4, 7$, and 10

Here, we can see an overwhelming amount of alignment between cells and their neighbors. Total neighbor alignment is indicated by cell trajectories colored in red. As J_S is increased, these plots suggest that the cells converge faster to a configuration of total polar alignment between neighbors. The direction of movement varies with each change in J_S and is dictated by the direction of minimum energy for the particular system.

5.4.4 Effects from groove alignment

To take a closer look at the effect of a grooved substrate on the motion of cells, we set $J_S = 0$ and vary J_G between 1 and 10. We produce four orientation plots at $J_G = 1, 4, 7$, and 10 shown in fig. 21.

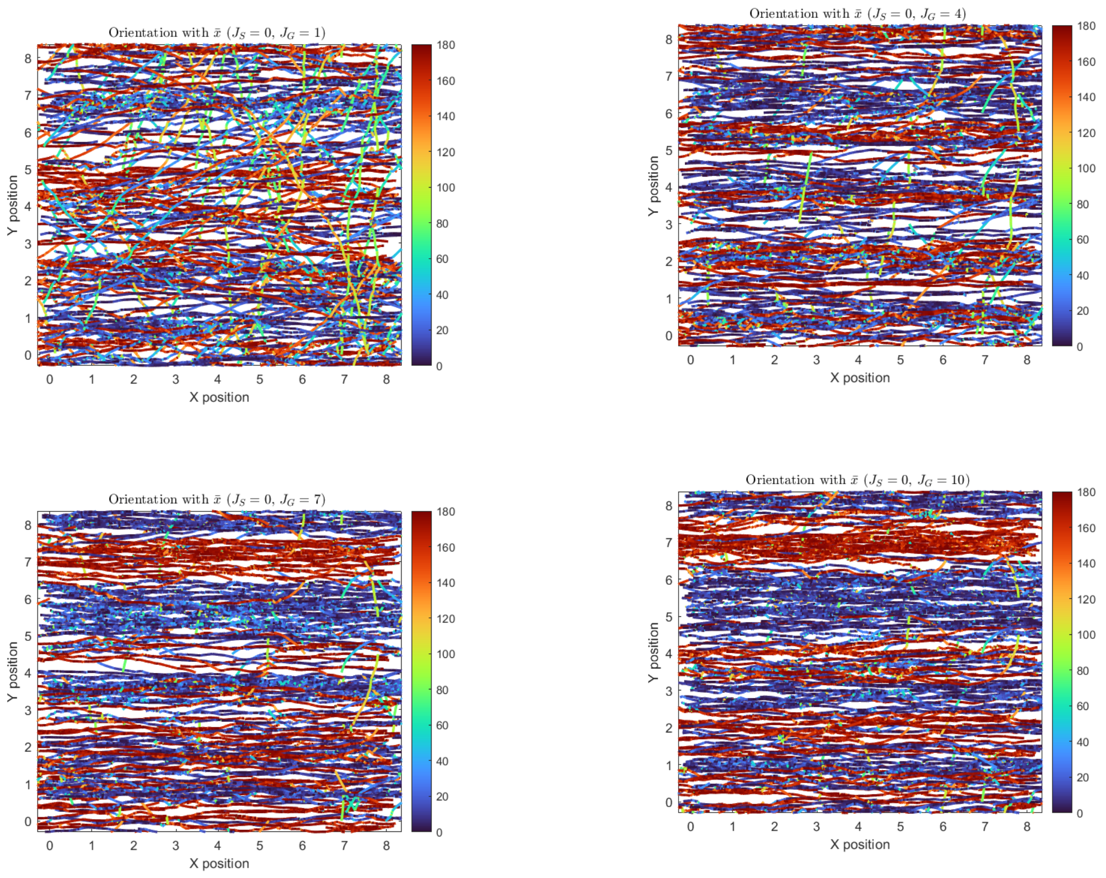


Figure 21: Orientation plots using $J_G = 0$ and $J_G = 1, 4, 7,$ and 10

These plots clearly elicit anti-parallel stream behavior. The trajectories in dark blue show cells moving in the direction of the positive x-axis and trajectories in dark red show cells aligned with the negative x-axis.

Increasing J_G emulates the effect of increasing the groove depth in the simulation. We conclude from the plots above, that varying the groove depth changes how quickly the system will converge to anti-parallel streams.

5.4.5 Larger simulation

For an even deeper analysis of groove alignment we look at a fluid state simulation with 400 cells. For this simulation, we use a groove alignment modulus of 10 and 0 neighbor alignment. Fig. 22 shows the evolution of the system as it goes from a perfect hexagonal configuration to a fluid state and finally to a rosettes moving in anti-parallel cell streams.

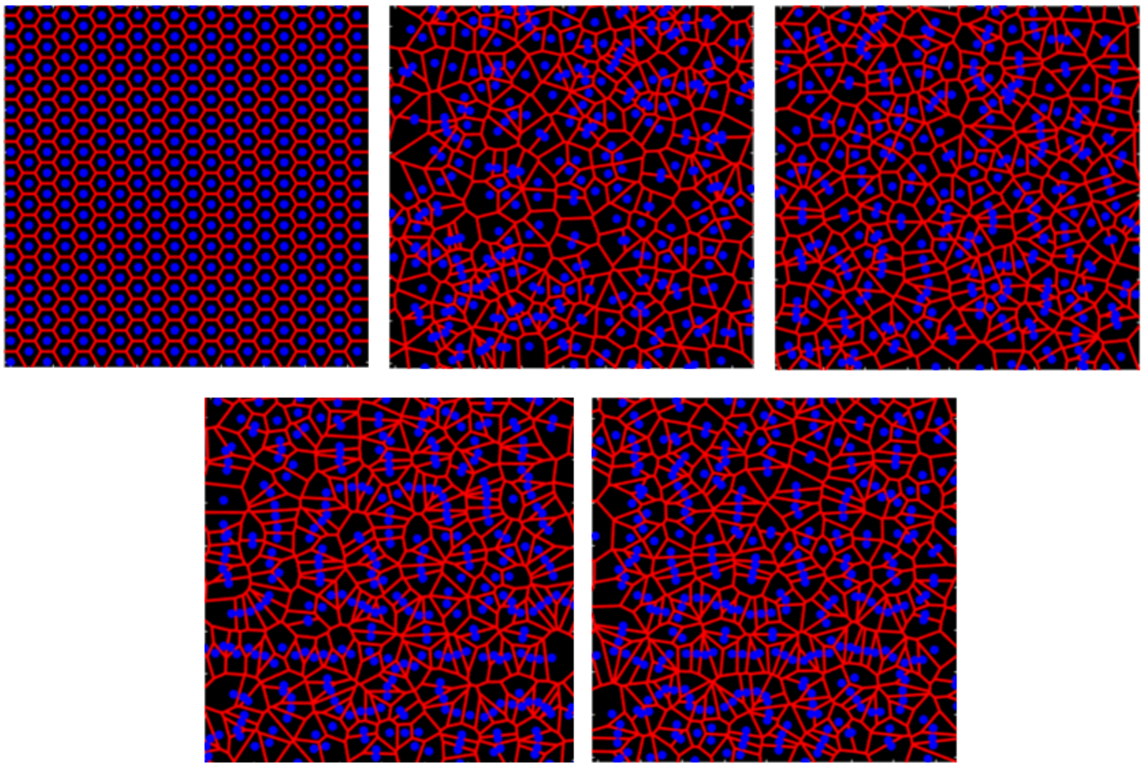


Figure 22: Fluid state simulation with 400 cells

We specifically analyze the effects of a grooved substrate for this system by, again, looking at the coordination and orientation plots (figure 23).

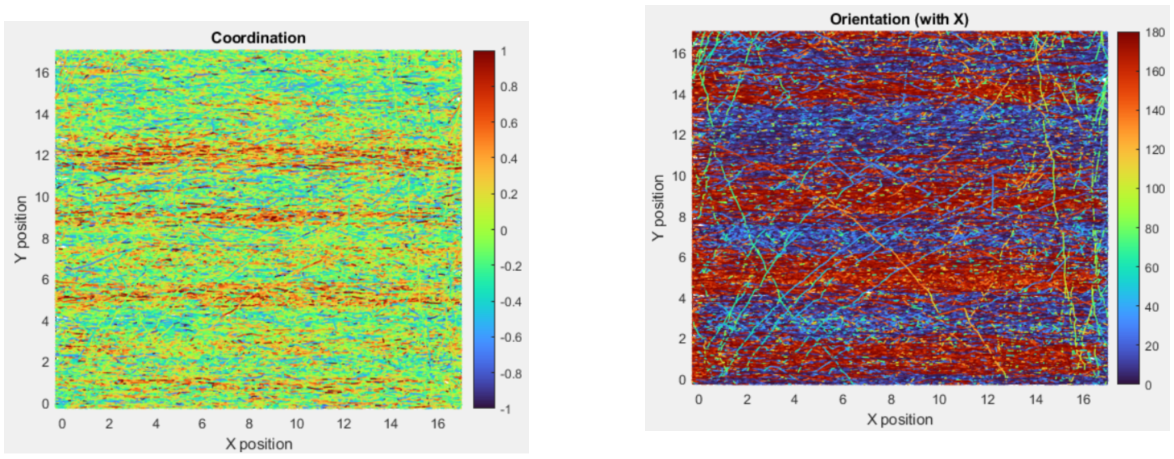


Figure 23: Coordination and orientation plots for fluid simulation with 400 cells ($J_S = 0$, $J_G = 10$)

Here, the modeled tissue exhibits clear cell-streams, emphasized by the overwhelming red and blue lines from the orientation plot. For comparison, we also show the experimental findings of (Leclech et al., 2022) where cell trajectories in an endothelial monolayer were tracked and colored according the same basic specifications as in our plots (fig. 24).

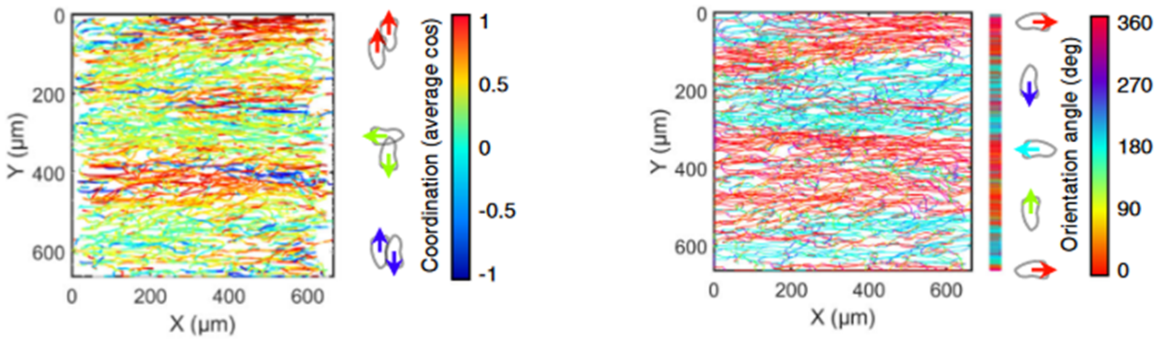


Figure 24: Experimental coordination and orientation plots from (Leclech et al., 2022)

In both fig. 22 and fig. 24, anti-parallel cell stream patterns were observed yielding remarkably similar coordination and orientation plots. This suggests that our model is able to accurately predict experimental behavior

6 Discussion

This project sets the groundwork for modeling confluent cell migration patterns in a particle dynamics framework LAMMPS via a vertex model. Our model partitions a system of given cell centers into cells using the properties of the Dual Voronoi Tessellation and applies forces onto cell centers in such a way that minimizes the energy of the system. The two mechanisms that drive cell behavior in our model are vertex model forces and forces of alignment. In this paper, we discuss our approach for implementing this model into LAMMPS and provide results for unjamming transitions, timestep convergence, and cell migration replication for a monolayer of endothelial cells. This section details the limitations of our current model and discusses areas of future work.

6.1 Limitations

Perhaps the most glaring limitation found in this project is the huge computational cost incurred from our implementation of the `fix customforce` function. The bulk of this cost stems from finding and storing the Delaunay triangles and their respective vertices at every iteration during simulation. This is an obvious hazard for future work because it makes it almost impossible to model the large and complex systems that are found in real biological tissues. The cost of our LAMMPS implementation also relies heavily on the LAMMPS neighbors cutoff threshold. This cutoff distance simply defines the range for which LAMMPS will search for the neighbors of a cell. This aspect plays a crucial role in vertex and DT matrix construction because any cell center outside this threshold cannot be recognized, resulting in compilation errors when vertices are missing. To avoid this, the cutoff can be set to a large value that is sure to capture every cell neighbor, but this comes at the cost of severe compilation hangups. In the future, we think the best approach to solve this would be to find a way to change this value dynamically as the simulation progresses because, in most cases, the cutoff can remain small.

Another limitation of this project has been the exclusive use of Forward Euler as a time integrator in our Brownian dynamics equation of motion. While this timestepping method is simple and arrives at straightforward solutions for deterministic ODEs, it also comes with well-known accuracy problems which could put the validity of some simulations in question. In the future, we suggest implementing a more modern timestepping scheme (such as Runge-Kutta), to avoid such accuracy issues and perhaps allow for larger timesteps to be used.

6.2 Future work

Besides tending to the aforementioned limitations, we propose some additional areas for future work that will make our model more complete. The first of these areas is to take full advantage of the parallel architecture of LAMMPS. Without utilizing this aspect of the simulator, larger and more complex simulations are potentially far out of reach. LAMMPS is specifically designed to handle running simulations in parallel by using the Message Passing Interface (MPI) parallel communication standard.

After parallelization has been implemented, another obvious emphasis of future work should be the implementation of other important cell functionalities including cell growth, apoptosis, and birth via cell division. Many previous vertex models have successfully modeled these features (Barton et al., 2017), so adding them to our model should be relatively straightforward.

Future work in solid state analysis is also a great way to emulate results from previous work. Such analyses would likely consist of simulating the effects of deformations onto tissue elasticity by measuring changes in the Young's modulus and the Poisson's ratio (Aziza, 2018). To conduct these investigations, it is first necessary to further research how virial stresses are applied in the LAMMPS framework.

Finally, we hope to eventually extend our model into three dimensions. The large limitation associated with this extension would be the translation of the mathematical concepts (Voronoi tessellation, Delaunay triangulation, force equation etc.) presented in this paper. LAMMPS itself is very adept at handling simulations in 3D, so once the tissue structure and necessary equations have been clearly mapped, a three dimensional implementation should follow with relative ease.

References

Alt, S., Ganguly, P., Salbreux, G., 2017. Vertex models: from cell mechanics to tissue morphogenesis. Philosophical Transactions of the Royal Society B: Biological Sciences 372, 20150520. URL: https://royalsocietypublishing.org/doi/10.1098/rstb.2015.0520, doi:10.1098/rstb.2015.0520.

Asnacios, A., Hamant, O., 2012. The mechanics behind cell polarity. Trends in Cell Biology 22, 584–591. URL: https://linkinghub.elsevier.com/retrieve/pii/S0962892412001444, doi:10.1016/j.tcb.2012.08.005.

Aziza, Fatma, M., 2018. Numerical Modelling of Confluent Cell Monolayers : Study of Tissue Mechanics and Morphogenesis URL: https://archive-ouverte.unige.ch/unige:106434, doi:10.13097/ARCHIVE-OUVERTE/UNIGE:106434. publisher: Université de Genève.

Barton, D.L., Henkes, S., Weijer, C.J., Sknepnek, R., 2017. Active Vertex Model for cell-resolution description of epithelial tissue mechanics. PLOS Computational Biology 13, e1005569. URL: https://dx.plos.org/10.1371/journal.pcbi.1005569, doi:10.1371/journal.pcbi.1005569.

Bi, D., Lopez, J.H., Schwarz, J.M., Manning, M.L., 2015. A density-independent rigidity transition in biological tissues. Nature Physics 11, 1074–1079. URL: http://www.nature.com/articles/nphys3471, doi:10.1038/nphys3471.

Blankenship, J.T., Backovic, S.T., Sanny, J.S.P., Weitz, O., Zallen, J.A., 2006. Multicellular rosette formation links planar cell polarity to tissue morphogenesis. Dev Cell 11, 459–470.

Chaté, H., Ginelli, F., Grégoire, G., Peruani, F., Raynaud, F., 2008. Modeling collective motion: variations on the vicsek model. The European Physical Journal B 64, 451–456. URL: https://doi.org/10.1140/epjb/e2008-00275-9, doi:10.1140/epjb/e2008-00275-9.

Farhadifar, R., Röper, J.C., Aigouy, B., Eaton, S., Jülicher, F., 2007. The Influence of Cell Mechanics, Cell-Cell Interactions, and Proliferation on Epithelial Packing. Current Biology 17, 2095–2104. URL: https://linkinghub.elsevier.com/retrieve/pii/S0960982207023342, doi:10.1016/j.cub.2007.11.049.

Fletcher, A.G., Cooper, F., Baker, R.E., 2017. Mechanocellular models of epithelial morphogenesis. Philosophical Transactions of the Royal Society B: Biological Sciences 372, 20150519. URL: https://royalsocietypublishing.org/doi/10.1098/rstb.2015.0519, doi:10.1098/rstb.2015.0519.

Fletcher, A.G., Osborne, J.M., Maini, P.K., Gavaghan, D.J., 2013. Implementing vertex dynamics models of cell populations in biology within a consistent computational framework. Progress in Biophysics and Molecular Biology 113, 299–326. URL: https://linkinghub.elsevier.com/retrieve/pii/S0079610713000989, doi:10.1016/j.pbiomolbio.2013.09.003.

Friedl, P., Gilmour, D., 2009. Collective cell migration in morphogenesis, regeneration and cancer. Nature Reviews Molecular Cell Biology 10, 445–457. URL: http://www.nature.com/articles/nrm2720, doi:10.1038/nrm2720.

Heisenberg, C.P., Bellaïche, Y., 2013. Forces in Tissue Morphogenesis and Patterning. Cell 153, 948–962. URL: https://linkinghub.elsevier.com/retrieve/pii/S0092867413005734, doi:10.1016/j.cell.2013.05.008.

Honda, H., 1978. Description of cellular patterns by dirichlet domains: The two-dimensional case. Journal of Theoretical Biology 72, 523–543. URL: https://www.sciencedirect.com/science/article/pii/0022519378903156, doi:https://doi.org/10.1016/0022-5193(78)90315-6.

Kursawe, J., Baker, R.E., Fletcher, A.G., 2017. Impact of implementation choices on quantitative predictions of cell-based computational models. Journal of Computational Physics 345, 752–767. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999117304412, doi:10.1016/j.jcp.2017.05.048.

Leclech, C., Gonzalez-Rodriguez, D., Villedieu, A., Lok, T., Déplanche, A.M., Barakat, A.I., 2022. Topography-induced large-scale antiparallel collective migration in vascular endothelium. Nature Communications 13, 2797. URL: https://www.nature.com/articles/s41467-022-30488-0, doi:10.1038/s41467-022-30488-0.

Lin, S.Z., Li, B., Feng, X.Q., 2017. A dynamic cellular vertex model of growing epithelial tissues. Acta Mechanica Sinica 33, 250–259. URL: http://link.springer.com/10.1007/s10409-017-0654-y, doi:10.1007/s10409-017-0654-y.

Milde, F., Tauriello, G., Haberkern, H., Koumoutsakos, P., 2014. SEM++: A particle model of cellular growth, signaling and migration. Computational Particle Mechanics 1, 211–227. URL: http://link.springer.com/10.1007/s40571-014-0017-4, doi:10.1007/s40571-014-0017-4.

Montell, D.J., 2008. Morphogenetic Cell Movements: Diversity from Modular Mechanical Properties. Science 322, 1502–1505. URL: https://www.science.org/doi/10.1126/science.1164073, doi:10.1126/science.1164073.

Nava-Sedeño, J.M., Voß-Böhme, A., Hatzikirou, H., Deutsch, A., Peruani, F., 2020. Modelling collective cell motion: are on- and off-lattice models equivalent? Philosophical Transactions of the Royal Society B: Biological Sciences 375, 20190378. URL: https://royalsocietypublishing.org/doi/10.1098/rstb.2019.0378, doi:10.1098/rstb.2019.0378.

Rozman, J., Krajnc, M., Ziherl, P., 2020. Collective cell mechanics of epithelial shells with organoid-like morphologies. Nature Communications 11, 3805. URL: http://www.nature.com/articles/s41467-020-17535-4, doi:10.1038/s41467-020-17535-4.

Sadati, M., Taheri Qazvini, N., Krishnan, R., Park, C.Y., Fredberg, J.J., 2013. Collective migration and cell jamming. Differentiation 86, 121–125. URL: https://linkinghub.elsevier.com/retrieve/pii/S0301468113000170, doi:10.1016/j.diff.2013.02.005.

Staple, D.B., Farhadifar, R., Röper, J.C., Aigouy, B., Eaton, S., Jülicher, F., 2010. Mechanics and remodelling of cell packings in epithelia. The European Physical Journal E 33, 117–127. URL: `http://link.springer.com/10.1140/epje/i2010-10677-0`, doi:`10.1140/epje/i2010-10677-0`.

Thompson, A.P., Aktulga, H.M., Berger, R., Bolintineanu, D.S., Brown, W.M., Crozier, P.S., in 't Veld, P.J., Kohlmeyer, A., Moore, S.G., Nguyen, T.D., Shan, R., Stevens, M.J., Tranchida, J., Trott, C., Plimpton, S.J., 2022. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. Comp. Phys. Comm. 271, 108171. doi:`10.1016/j.cpc.2021.108171`.

Trichas, G., Smith, A.M., White, N., Wilkins, V., Watanabe, T., Moore, A., Joyce, B., Sugnaseelan, J., Rodriguez, T.A., Kay, D., Baker, R.E., Maini, P.K., Srinivas, S., 2012. Multi-cellular rosettes in the mouse visceral endoderm facilitate the ordered migration of anterior visceral endoderm cells. PLOS Biology 10, 1–14. URL: `https://doi.org/10.1371/journal.pbio.1001256`, doi:`10.1371/journal.pbio.1001256`.

Van Liedekerke, P., Palm, M.M., Jagiella, N., Drasdo, D., 2015. Simulating tissue mechanics with agent-based models: concepts, perspectives and some novel results. Computational Particle Mechanics 2, 401–444. URL: `http://link.springer.com/10.1007/s40571-015-0082-3`, doi:`10.1007/s40571-015-0082-3`.

Wagner, R.J., Vernerey, F.J., 2022. Computational exploration of treadmilling and protrusion growth observed in fire ant rafts. PLOS Computational Biology 18, 1–32. URL: `https://doi.org/10.1371/journal.pcbi.1009869`, doi:`10.1371/journal.pcbi.1009869`.

Wyczalkowski, M.A., Chen, Z., Filas, B.A., Varner, V.D., Taber, L.A., 2012. Computational models for mechanics of morphogenesis. Birth Defects Research Part C: Embryo Today: Reviews 96, 132–152. URL: `https://onlinelibrary.wiley.com/doi/10.1002/bdrc.21013`, doi:`10.1002/bdrc.21013`.

Yan, L., Bi, D., 2019. Multicellular Rosettes Drive Fluid-solid Transition in Epithelial Tissues. Physical Review X 9, 011029. URL: `https://link.aps.org/doi/10.1103/PhysRevX.9.011029`, doi:`10.1103/PhysRevX.9.011029`.

Zhang, S.C., Wernig, M., Duncan, I.D., Brüstle, O., Thomson, J.A., 2001. In vitro differentiation of transplantable neural precursors from human embryonic stem cells. Nature Biotechnology 19, 1129–1133. URL: `https://doi.org/10.1038/nbt1201-1129`, doi:`10.1038/nbt1201-1129`.