Technische Universität Dresden • Faculty of Mathematics

Derivation and study of a non-confluent model for deformable cells

Master's thesis

to obtain the second degree

Master of Science
(*M.Sc.*)

written by

Tim Vogel

(born on June 9, 2002 in Finsterwalde)

Day of submission: September 1, 2025

Supervised by Jun.-Prof. Dr. Markus Schmidtchen
(Institute of Scientific Computing)

Contents

1 TODOs

Cell dynamics - SDE

- introduce the energies with the k: $E = 1/k|...|^k$ and also adapt the forces accordingly
- watch out to use only the needed indices in the notation of energies and forces (a force function should get the considered cell as an argument, but itself should be independent on the according index)
- add computations of the 'neighboring terms' in the proofs of the forces
- write down nicely, what the SDE is, that we would like to solve. Is it $\frac{\mathrm{d}\vec{C}}{\mathrm{d}t} = ...$, or $\frac{\mathrm{d}\vec{v}_j^i}{\mathrm{d}t} = ...$ for $1 \leqslant i \leqslant N_C$ and $1 \leqslant j \leqslant N_V$? Maybe introduce both. Biggest Issue: What is the deterministic part (F). Always write down the dimensions of the arguments and where F maps into. Write down the force functions accordingly, so we can just say: $(F) = a_{area}F_{aray} + a_{edge}F_{edge} + ...$ Do we need $(F)(\vec{C}, C_i)$ or how is it correct?
- Introduce new billiardBounceOverlapDegeneration.
- Introduce hardness parameter.

Introduction

- KEEP BACHELOR INTRO
- VERTEX BASED MODELS WERDEN FÜR CONFLUENT MODELS GEMACHT - DAVON MÖCHTEN WIR WEGKOMMEN
- SEE Jamming of Deformable Polygons.pdf 17
- Jamming of Deformable Polygons Supplemental Materials
- LOOK FOR SCALING FACTORS
- WRITE DOWN USE CASES
- IN CONFIGS WHERE THE CELLS SHOULD NOT BE GLUED TOGETHER
- WE NEED THE NON CONFLUENT CELL MODEL + WAS MARKUS GESENDET HAT
- ALSO MENTION AXEL VOIGTS PAPER IF HE IS 2ND CONTROLLEUR

Density comps

- think about density pde. Remember edge computation with edge energy and 2 vertices cells (photo!)
- read robins BT, learn how this shit works

- do the edge computation for N_v vertices
- then do it for area energy

formal things

- add left(, right) where it is necessary
- control indices

2 DF cell model

The following two sections are a recap of the DF cell model and its dynamics that were introduced in my Bachelor's thesis [Vog23].
We are considering cells in the two dimensional space $\mathbb{R}^2$. Here, cells are considered to be polygons.

Definition 2.1. Polygon
A polygon is a closed geometric figure in $\mathbb{R}^2$, constructed by joining a finite number of straight line segments end to end. It can be described by a sequence of its vertices $(\vec{v}_1, \dots, \vec{v}_N)$. The following properties characterise a polygon:

1. A polygon is **simple** if no two line segments cross each other.
2. A polygon has a **positive orientation** if the vertices are ordered counter-clockwise.
3. A polygon has a **negative orientation** if the vertices are ordered clockwise.

Having established this definition, we are now ready to define our cell model.

Definition 2.2. Discrete form (DF)
A cell in its discrete form (**DF**) is given by an ordered sequence of its vertices $C = (\vec{v}_1, \dots, \vec{v}_N)$ if the resulting polygon when connecting every vertex with its neighbours and $\vec{v}_1$ with $\vec{v}_N$ is simple and positively orientated. We set $\vec{v}_{N+1} = \vec{v}_1$ and $\vec{v}_0 = \vec{v}_N$ to enable periodic indexing, which simplifies the computation of the upcoming forces a lot.

In this thesis, DF cells may also be called discrete cells. In our model, the cell vertices are denoted by $\vec{v}$. Thus, the character v refers to vertex positions and not to velocity. The term velocity is not used throughout this thesis as vertex dynamics are entirely given by the upcoming forces and a cell wise computed Brownian motion.

The next step is to describe the setup of a DF simulation.

Definition 2.3. DF simulation
A DF simulation considers $N_C \in \mathbb{N}$ cells. Each cell has the same amount of $N_V \in \mathbb{N}$ vertices. Thus, the notation of all cells and their vertices is given by

$$C^i = (\vec{v}_1^{\,i}, \dots, \vec{v}_{N_V}^{\,i}), \quad 1 \leqslant i \leqslant N_C.$$

The complete set of all cells is represented by

$$\vec{C} = (C^1, \dots, C^{N_C}),$$

which also contains all vertices from all cells.
The simulation's dynamics are defined on all cell vertices via the stochastic differential equation (SDE):

$$\mathrm{d}\vec{v}_j^{\,i}(\vec{C}) = \mathbf{F}_j^i(\vec{C})\mathrm{d}t + \sqrt{2D}\mathrm{d}\vec{B}^{\,i}, \quad 1 \leqslant i \leqslant N_C, \quad 1 \leqslant j \leqslant N_V.$$

where $\mathbf{F}_j^i$ describes the total interaction force on vertex $\vec{v}_j^{\,i}$ caused by the current cell system $\vec{C}$ and $\sqrt{2D}\mathrm{d}\vec{B}^{\,i}$ models the two dimensional standard Brownian motion of cell i with diffusion coefficient D. Note, that all vertices of cell i perform the same Brownian motion such that the whole cell i moves in the direction of $\mathrm{d}\vec{B}^{\,i}$.

The simulation domain is always a square around the origin that is defined by $L > 0$ via

$$\Omega_L = [-L, L]^2.$$

How the interaction force $\mathbf{F}$ can be modelled will be shown the next chapter.

3 DF model dynamics

We characterise the interaction force $\mathbf{F}$ as the sum of gradient flows of energies.
A gradient flow describes how a system changes over time in a way that always reduces a given energy $E(\vec{C})$. To obtain the gradient flow of this energy on vertex $\vec{v}$, we must add the term $-\nabla_{\vec{v}} E(\vec{C})$ to $\mathbf{F}$. Since all our energy terms are positive, the lowest possible value is zero. So, the gradient flow moves the system step by step toward this minimum, always trying to decrease the energy until, ideally, it reaches zero. This is how we guide the motion of our cells: by letting them follow the gradient flow of each energy so that their shapes and vertex positions gradually adjust to reduce the total energy.
In [Vog23], the area, edge, interior angle, and overlap energies were introduced. The first three energies are responsible for maintaining the shape of each cell. All of these three according forces act on each cell in a vacuum based only on its own current cell shape.
Unlike in [Vog23], where each cell was assigned an individual desired state, we now assume a common desired state for all cells. This simplification allows for a more controlled analysis of the system's deformability and its influence on the collective dynamics. We assume that all cells are initially given in their desired states in order to prevent system instabilities right from the beginning.
Additionally, we introduce slight modifications to the energy formulation: rather than being defined locally on vertices or edges, the energies are now defined over entire cells. This adjustment provides a more coherent basis for deriving cell-level forces and ensures consistency with the global dynamic framework introduced in this study.
Interactions between different cells just arise from the overlap force, which acts to resolve overlaps and to prevent cell interpenetration. In the process of resolving overlaps, the shape of the cells will change. Once the overlap is resolved, the first three forces act to restore the cell's original shape.
The central question we aim to investigate in this thesis is how the deformability of individual cells influences the overall diffusivity of the cell system. But first, let us introduce each of the mentioned forces.
We define our energies as

$$E_k(x) = \frac{1}{k} |x_{\text{desired}} - x_{\text{current}}|^k,$$

where $k \in \mathbb{N}_{\geqslant 1}$ is a positive integer parameter specific to each energy term. Using different values of k allows us to model various types of energies and their corresponding forces, resulting in distinct dynamical behaviors that reflect different aspects of cell physics.
In order to compute the forces arising from these energy functions, we require the gradient ∇E. This leads us to compute derivatives of the form

$$\frac{\mathrm{d}}{\mathrm{d}x} |x|^k.$$

While $|x|^k$ is not classically differentiable at $x = 0$, it is weakly differentiable for all $k \in \mathbb{N}_{\geqslant 1}$. There exists a locally integrable function

$$x \mapsto k \operatorname{sgn}(x)|x|^{k-1} \in L^1_{loc}(\mathbb{R}),$$

such that for all $\phi \in C^\infty_C(\mathbb{R})$:

$$\begin{aligned}
\int_{\mathbb{R}} |x|^k \phi'(x) \mathrm{d}x &= \int_{\mathbb{R}_{\geqslant 0}} x^k \phi'(x) \mathrm{d}x + \int_{\mathbb{R}_{<0}} (-x)^k \phi'(x) \mathrm{d}x \\
&= [x^k \phi(x)]_0^\infty - \int_{\mathbb{R}_{\geqslant 0}} k x^{k-1} \phi(x) \mathrm{d}x + [(-x)^k \phi(x)]_0^\infty - \int_{\mathbb{R}_{<0}} k(-x)^{k-1} \phi(x) \mathrm{d}x \\
&= - \int_{\mathbb{R}_{\geqslant 0}} k x^{k-1} \phi(x) \mathrm{d}x - \int_{\mathbb{R}_{<0}} k(-x)^{k-1} \phi(x) \mathrm{d}x \\
&= - \int_{\mathbb{R}} k \operatorname{sgn}(x)|x|^{k-1} \phi(x) \mathrm{d}x.
\end{aligned}$$

Thus, $x \mapsto k \operatorname{sgn}(x)|x|^{k-1}$ is the weak derivative of $x \mapsto |x|^k$. We will use this weak derivative for all of our force computations.

3.1 Area force

The area force is designed to maintain each cell's area close to a preferred target value. In order to compute a cells area, which is the area of a positively orientated polygon, we can use the Shoelace formula from [Sho14].

Proposition 3.1. ***Shoelace formula for DF cells***
Let $C = (\vec{v}_1, \ldots, \vec{v}_N)$ be a DF cell with $\vec{v}_j = (v_j^x, v_j^y)^T$ for $j = 1, \ldots, N$. We determine the area A_C of C by applying the Shoelace formula

$$A_C = \tfrac{1}{2} \sum_{j=1}^{N} (v_j^x v_{j+1}^y - v_{j+1}^x v_j^y),$$

where $\vec{v}_{N+1} = \vec{v}_1$.
Proof.
An illustration supporting the proof is provided in 1, which is where the idea of the proof comes from. Without loss of generality, we may assume that all coordinates are positive. If this is not initially the case, the entire polygon can be translated into the positive quadrant without affecting its area.
For each $1 \leqslant j \leqslant N$ the edge $\overline{\vec{v}_j \, \vec{v}_{j+1}}$ is associated with the area T_j of the trapeze that arises when connecting the line segment vertically with the x axis. The signed trapeze area of T_j can be computed with

$$T_j = \tfrac{1}{2}(v_j^y + v_{j+1}^y)(v_j^x - v_{j+1}^x).$$

The area T_j has a positive sign if $v_j^x \geqslant v_{j+1}^x$ (green arrow in Figure 1) and a negative sign otherwise (red arrow). As depicted in the figure, the negatively signed areas precisely cancel the excess portions that would result from summing only the positively signed trapezoids. Thus the total polygon's area is equal to the sum of all trapezes

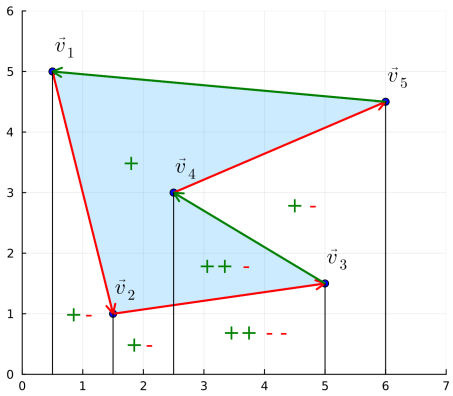

Figure 1: This figure shows a geometrical interpretation of the shoelace formula. The green arrows, which point from right to left, represent positive trapezoidal areas that contribute positively to the total area of the polygon. In contrast, the red arrows point from left to right and represent negative areas that are subtracted in the computation. The vertical black lines divide the plot into subregions. Within each subregion, green arrows are counted with plus signs and red arrows with minus signs. We observe that the subregions lying outside the polygon contain an equal number of plus and minus signs, indicating that their net contribution to the area is zero. In contrast, the subregions inside the polygon always have one more plus sign than minus signs, meaning their area is counted exactly once in the total. Overall, this illustrates that the method correctly computes the area of the polygon. Source: [Sho22]

$$A_C = \sum_{j=1}^{N} T_j = \tfrac{1}{2} \sum_{j=1}^{N} (v_j^y + v_{j+1}^y)(v_j^x - v_{j+1}^x) = \tfrac{1}{2} \sum_{j=1}^{N} (v_j^x v_{j+1}^y - v_{j+1}^x v_j^y).$$

□

With the Shoelace formula we are able to easily compute all cell areas at all times in the simulation. This enables us to implement the gradient flow over the area energy.

Definition 3.2. Area energy
The energy $A_k : (\mathbb{R}^2)^{N_V} \to \mathbb{R}_{\geqslant 0}$ for $k \in \mathbb{N}_{\geqslant 1}$, used to keep the cells at a constant volume, reads

$$A_k(C) = \frac{1}{k} |A_C - A_d|^k, \tag{1}$$

where A_d is the desired cell area of all cells and A_C is the current area of cell C.
To maintain the cell area during the simulation, we evaluate the gradient flow of the area energy which indicates the direction of motion for each vertex for preserving the cell area.

Proposition 3.3. ***Area force***
The area force $F_k^{(A)} : (\mathbb{R}^2)^{N_V} \to (\mathbb{R}^2)^{N_V}$ *that gets applied on cell* C *is given by*

$$F_k^{(A)}(C) = -(\nabla_{\vec{v}_1} A_k(C), \dots, \nabla_{\vec{v}_{N_V}} A_k(C))^T,$$

where the gradient $\nabla_{\vec{v}_j} A_k(C)$ *with respect to* $\vec{v}_j = (v_j^x, v_j^y)^T$ *is given by*

$$\nabla_{\vec{v}_j} A_k(C) = \frac{1}{2} \operatorname{sgn}(A_C - A_d) |A_C - A_d|^{k-1} \begin{pmatrix} v_{j+1}^y - v_{j-1}^y \\ v_{j-1}^x - v_{j+1}^x \end{pmatrix}, \tag{2}$$

for all $1 \leqslant j \leqslant N_V$.

Proof.
Choose $1 \leqslant j \leqslant N_V$.

$$\begin{aligned}
\nabla_{\vec{v}_j} A_k(C) &= \frac{1}{k} \nabla_{\vec{v}_j} |A_C - A_d|^k \\
&= \operatorname{sgn}(A_C - A_d) |A_C - A_d|^{k-1} \nabla_{\vec{v}_j} (A_C - A_d) \\
&= \operatorname{sgn}(A_C - A_d) |A_C - A_d|^{k-1} \nabla_{\vec{v}_j} A_C \\
&= \operatorname{sgn}(A_C - A_d) |A_C - A_d|^{k-1} \nabla_{\vec{v}_j} \left(\frac{1}{2} \sum_{k=1}^{N} (v_k^x v_{k+1}^y - v_{k+1}^x v_k^y) \right) \\
&= \frac{1}{2} \operatorname{sgn}(A_C - A_d) |A_C - A_d|^{k-1} \begin{pmatrix} \partial_{v_j^x} (v_j^x v_{j+1}^y - v_j^x v_{j-1}^y) \\ \partial_{v_j^y} (v_{j-1}^x v_j^y - v_{j+1}^x v_j^y) \end{pmatrix} \\
&= \frac{1}{2} \operatorname{sgn}(A_C - A_d) |A_C - A_d|^{k-1} \begin{pmatrix} v_{j+1}^y - v_{j-1}^y \\ v_{j-1}^x - v_{j+1}^x \end{pmatrix}
\end{aligned}$$

Remember that A_d *is just an independent constant.* □

It is also valid to write $F_j^{(A)}(\vec{C})$ instead of $F_j^{(A)}(C)$, since C is included in $\vec{C}$.
Figure 2 illustrates how the area force acts on a cell to either expand or contract it toward the desired target area.

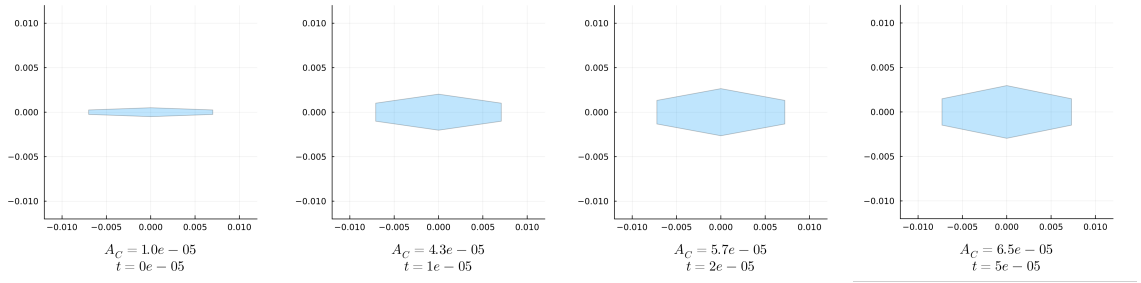

Figure 2: Click: *here*, to see the animation as a gif.

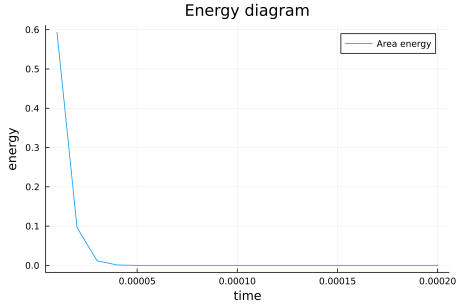

Figure 3

3.2 Edge force

The next force we would like to model is the edge force. It acts on the cells' edges and aims to maintain their lengths. We define the edge $1 \leqslant j \leqslant N_V$ as

$$e_j = \overline{\vec{v}_j \, \vec{v}_{j+1}}$$

and we use the operator

$$E_C^j = \|\vec{v}_j - \vec{v}_{j+1}\|_2$$

to compute the length of the edge.
The according energy for this edge is:

Definition 3.4. Edge energy
The energy $E_k : (\mathbb{R}^2)^{N_V} \to \mathbb{R}_{\geqslant 0}$, used to keep the edges at a constant length, reads

$$E_k(C) = \sum_{j=1}^{N_V} \frac{1}{k} |E_C^j - E_d^j|^k, \tag{3}$$

where E_C^j is the current edge length and E_d^j is the desired edge length of edge j.

Since each vertex $\vec{v}_j$ influences exactly the edge lengths of the edges e_j and e_{j-1}, we get the total edge force on $\vec{v}_j$ with:

Proposition 3.5. ***Edge force***

The edge force $F_k^{(E)} : (\mathbb{R}^2)^{N_V} \to (\mathbb{R}^2)^{N_V}$ *that gets applied on cell* C *is given by*

$$F_k^{(E)}(C) = -(\nabla_{\vec{v}_1} E_k(C), \ldots, \nabla_{\vec{v}_{N_V}} E_k(C))^T,$$

where the gradient $\nabla_{\vec{v}_j} E_k(C)$ *with respect to* $\vec{v}_j = (v_j^x, v_j^y)^T$ *is given by*

$$\begin{aligned} \nabla_{\vec{v}_j} E_k(C) = \operatorname{sgn}(E_C^{j-1} - E_d^{j-1}) \frac{|E_C^{j-1} - E_d^{j-1}|^{k-1}}{E_C^{j-1}} \begin{pmatrix} v_j^x - v_{j-1}^x \\ v_j^y - v_{j-1}^y \end{pmatrix} \\ + \operatorname{sgn}(E_C^{j} - E_d^{j}) \frac{|E_C^{j} - E_d^{j}|^{k-1}}{E_C^{j}} \begin{pmatrix} v_j^x - v_{j+1}^x \\ v_j^y - v_{j+1}^y \end{pmatrix} \end{aligned} \tag{4}$$

for all $1 \leqslant j \leqslant N_V$.

Proof.

$$\begin{aligned} \nabla_{\vec{v}_j} E(C) &= \nabla_{\vec{v}_j} \sum_{j=1}^{N_V} \frac{1}{k} |E_C^j - E_d^j|^k \\ \nabla_{\vec{v}_j} E(C) &= \nabla_{\vec{v}_j} \sum_{j=1}^{N_V} \frac{1}{k} |E_C^j - E_d^j|^k \\ &= \frac{1}{k} \nabla_{\vec{v}_j} |E_C^{j-1} - E_d^{j-1}|^k + \frac{1}{k} \nabla_{\vec{v}_j} |E_C^j - E_d^j|^k \end{aligned}$$

For the first summand, we can compute

$$\begin{aligned}
\frac{1}{k}\nabla_{\vec{v}_j}|E_C^{j-1}-E_d^{j-1}|^k &= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})|E_C^{j-1}-E_d^{j-1}|^{k-1}\nabla_{\vec{v}_j}(E_C^{j-1}-E_d^{j-1}) \\
&= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})|E_C^{j-1}-E_d^{j-1}|^{k-1}\nabla_{\vec{v}_j}E_C^{j-1} \\
&= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})|E_C^{j-1}-E_d^{j-1}|^{k-1}\cdot \\
&\qquad \nabla_{\vec{v}_j}\left[(v_{j-1}^x-v_j^x)^2+(v_{j-1}^y-v_j^y)^2\right]^{\frac{1}{2}} \\
&= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})|E_C^{j-1}-E_d^{j-1}|^{k-1}\cdot \\
&\qquad \left(\frac{1}{2E_C^{j-1}}\nabla_{\vec{v}_j}\left[(v_{j-1}^x-v_j^x)^2+(v_{j-1}^y-v_j^y)^2\right]\right) \\
&= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})|E_C^{j-1}-E_d^{j-1}|^{k-1}\cdot \\
&\qquad \nabla_{\vec{v}_j}\left[(v_{j-1}^x-v_j^x)^2+(v_{j-1}^y-v_j^y)^2\right]^{\frac{1}{2}} \\
&= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})|E_C^{j-1}-E_d^{j-1}|^{k-1}\cdot \\
&\qquad \left(\frac{1}{2E_C^{j-1}}\nabla_{\vec{v}_j}\left[(v_{j-1}^x-v_j^x)^2+(v_{j-1}^y-v_j^y)^2\right]\right) \\
&= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})\frac{|E_C^{j-1}-E_d^{j-1}|^{k-1}}{2E_C^{j-1}}\begin{pmatrix}\partial_{v_j^x}(v_{j-1}^x-v_j^x)^2\\ \partial_{v_j^y}(v_{j-1}^y-v_j^y)^2\end{pmatrix} \\
&= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})\frac{|E_C^{j-1}-E_d^{j-1}|^{k-1}}{2E_C^{j-1}}\begin{pmatrix}-2(v_{j-1}^x-v_j^x)\\ -2(v_{j-1}^y-v_j^y)\end{pmatrix} \\
&= \operatorname{sgn}(E_C^{j-1}-E_d^{j-1})\frac{|E_C^{j-1}-E_d^{j-1}|^{k-1}}{E_C^{j-1}}\begin{pmatrix}v_j^x-v_{j-1}^x\\ v_j^y-v_{j-1}^y\end{pmatrix}
\end{aligned}$$

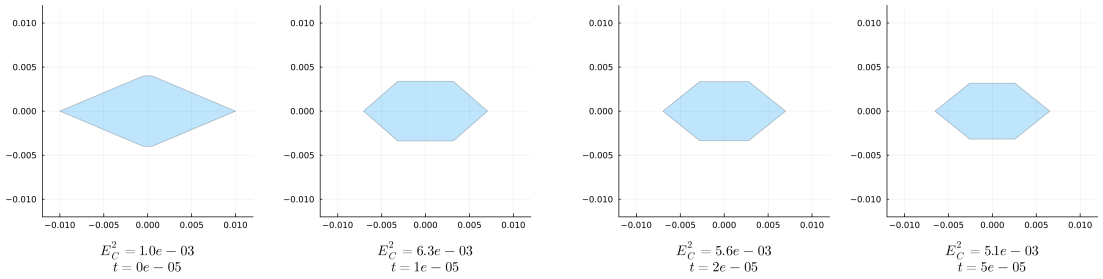

Figure 4: Click: *here*, to see the animation as a gif.

For the second summand, we get:

$$\begin{aligned}
\frac{1}{k}\nabla_{\vec{v}_j}|E_C^j - E_d^j|^k &= \operatorname{sgn}(E_C^j - E_d^j)|E_C^j - E_d^j|^{k-1}\nabla_{\vec{v}_j}E_C^j \\
&= \operatorname{sgn}(E_C^j - E_d^j)|E_C^j - E_d^j|^{k-1}\cdot \\
&\qquad \left(\frac{1}{2E_C^j}\nabla_{\vec{v}_j}[(v_j^x - v_{j+1}^x)^2 + (v_j^y - v_{j+1}^y)^2]\right) \\
&= \operatorname{sgn}(E_C^j - E_d^j)|E_C^j - E_d^j|^{k-1}\cdot \\
&\qquad \left(\frac{1}{2E_C^j}\nabla_{\vec{v}_j}[(v_j^x - v_{j+1}^x)^2 + (v_j^y - v_{j+1}^y)^2]\right) \\
&= \operatorname{sgn}(E_C^j - E_d^j)\frac{|E_C^j - E_d^j|^{k-1}}{2E_C^j}\begin{pmatrix}\partial_{v_j^x}(v_j^x - v_{j+1}^x)^2 \\ \partial_{v_j^y}(v_j^y - v_{j+1}^y)^2\end{pmatrix} \\
&= \operatorname{sgn}(E_C^j - E_d^j)\frac{|E_C^j - E_d^j|^{k-1}}{2E_C^j}\begin{pmatrix}2(v_j^x - v_{j+1}^x) \\ 2(v_j^y - v_{j+1}^y)\end{pmatrix} \\
&= \operatorname{sgn}(E_C^j - E_d^j)\frac{|E_C^j - E_d^j|^{k-1}}{E_C^j}\begin{pmatrix}v_j^x - v_{j+1}^x \\ v_j^y - v_{j+1}^y\end{pmatrix}
\end{aligned}$$

□

An isolated application of the edge force can be seen in Figure 4.

3.3 Interior angle force

The combined application of the area and edge forces revealed instabilities in unfavorable configurations, where self-intersections of the cell edges occurred. Simulations without this energy sometimes can also result in constrictions at certain vertices, where the interior angle approaches 360°. To address this issue, we introduce the interior angle energy.
The first challenge is to consistently determine the interior angle at a given vertex throughout the simulation. Although we could apply the law of cosines and use arccos to compute the angle, this method would suffer from poor stability as the

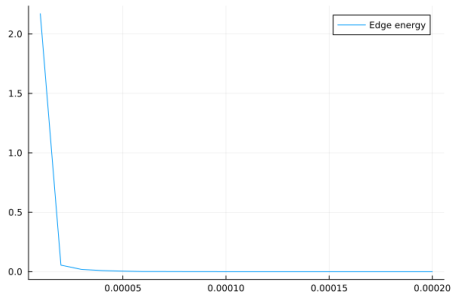

Figure 5

angle approaches 180°. A better alternative is to use the arctan2 function, as it remains reliably stable at all angles.

Definition 3.6. arctan2
The function

$$\text{arctan2} : \mathbb{R}^2/\{0\} \to (-\pi, \pi]$$

is defined by:

$$\text{arctan2}(\vec{v}) = \begin{cases} \arctan(\frac{v^y}{v^x}) & v^x > 0 \\ \arctan(\frac{v^y}{v^x}) + \pi & v^x < 0, v^y > 0 \\ \arctan(\frac{v^y}{v^x}) - \pi & v^x < 0, v^y < 0 \\ \pi & v^x < 0, v^y = 0 \\ \frac{\pi}{2} & v^x = 0, v^y > 0 \\ -\frac{\pi}{2} & v^x = 0, v^y < 0 \end{cases}.$$

The arctan2($\vec{v}$) function computes the angle of a vector $\vec{v}$ with respect to the positive x axis.
With this, we can compute the angles

$$\begin{aligned} \theta_1 &= \text{arctan2}(\vec{v}_{j-1} - \vec{v_j}), \\ \theta_2 &= \text{arctan2}(\vec{v}_{j+1} - \vec{v_j}) \end{aligned}$$

between the positive x axis and the vectors from $\vec{v}_j$ to its neighboring vertices $\vec{v}_{j-1}$ and $\vec{v}_{j+1}$. We get the searched angle at $\vec{v}_j$ by subtracting $\theta_1 - \theta_2$. To ensure that the angle lies within the interval $[0, 2\pi)$, we use the modulo operator $[\cdot]_{[0,2\pi)}$, which repeatedly adds or subtracts 2π from the angle until it falls within the desired range. Thus, our interior angle operator is:

$$I_C^j = [\text{arctan2}(\vec{v}_{j-1} - \vec{v_j}) - \text{arctan2}(\vec{v}_{j+1} - \vec{v_j})]_{[0,2\pi)}.$$

With that, we can define our interior angle energy.

Definition 3.7. Interior angle energy
The energy $I : (\mathbb{R}^2)^{N_V} \to \mathbb{R}_{\geqslant 0}$ associated with preserving the cell interior angles is

given by

$$I_k(C) = \sum_{j=1}^{N_V} \frac{1}{k} |I_C^j - I_d^j|^k, \tag{5}$$

where I_d^j is the desired interior angle at vertex j and I_C^j is the current interior angle at vertex j of the considered cell.

We continue by computing the resulting force. The arctan2 function is partly defined and not truly differentiable. We still want to compute a gradient to use it for our interior angle force. It is

$$\operatorname{arctan2}(\vec{v}) = \arctan\left(\frac{v^y}{v^x}\right) + \text{constant}$$

almost everywhere, just not on areas with measure zero. We just compute the gradient of $\arctan(\frac{v^y}{v^x})$ instead.
Another problem is the modulo operator $[\cdot]_{[0,2\pi)}$, which is not differentiable at the interval limits. However, we just neglect the modulo operator as it does not affect the dynamics of the gradient.

Proposition 3.8. ***Interior angle force***

The interior angle force $F_k^{(I)} : (\mathbb{R}^2)^{N_V} \to (\mathbb{R}^2)^{N_V}$ that gets applied on cell C is given by

$$F_k^{(I)}(C) = -(\nabla_{\vec{v}_1} I_k(C), \ldots, \nabla_{\vec{v}_{N_V}} I_k(C))^T,$$

where the gradient $\nabla_{\vec{v}_j} I_k(C)$ with respect to $\vec{v}_j = (v_j^x, v_j^y)^T$ is given by

$$\begin{aligned}
\nabla_{\vec{v}_j} I_k(C) = {} & \operatorname{sgn}(I_C^{j-1} - I_d^{j-1}) |I_C^{j-1} - I_d^{j-1}|^{k-1} \left(-\frac{1}{\|\vec{v}_j - \vec{v}_{j-1}\|_2^2} \begin{pmatrix} v_{j-1}^y - v_j^y \\ v_j^x - v_{j-1}^x \end{pmatrix} \right) \\
& + \operatorname{sgn}(I_C^j - I_d^j) |I_C^j - I_d^j|^{k-1} \left(\frac{1}{\|\vec{v}_{j-1} - \vec{v}_j\|_2^2} \begin{pmatrix} v_{j-1}^y - v_j^y \\ v_j^x - v_{j-1}^x \end{pmatrix} \right. \\
& \qquad \left. - \frac{1}{\|\vec{v}_{j+1} - \vec{v}_j\|_2^2} \begin{pmatrix} v_{j+1}^y - v_j^y \\ v_j^x - v_{j+1}^x \end{pmatrix} \right) \\
& + \operatorname{sgn}(I_C^{j+1} - I_d^{j+1}) |I_C^{j+1} - I_d^{j+1}|^{k-1} \left(\frac{1}{\|\vec{v}_j - \vec{v}_{j+1}\|_2^2} \begin{pmatrix} v_{j+1}^y - v_j^y \\ v_j^x - v_{j+1}^x \end{pmatrix} \right)
\end{aligned} \tag{6}$$

for all $1 \leqslant j \leqslant N_V$.

Proof.
We are looking for

$$\nabla_{\vec{v}_j} I_k(C)$$

Vertex $\vec{v}_j$ impacts the interior angles at $\vec{v}_{j-1}$, $\vec{v}_j$ and $\vec{v}_{j+1}$. Thus, we get

$$\begin{aligned}\nabla_{\vec{v}_j} I_k(C) &= \nabla_{\vec{v}_j} \sum_{j=1}^{N_V} \frac{1}{k} |I_d^j - I_C^j|^k \\ &= \nabla_{\vec{v}_j} \frac{1}{k} |I_d^{j-1} - I_C^{j-1}|^k + \nabla_{\vec{v}_j} \frac{1}{k} |I_d^j - I_C^j|^k + \nabla_{\vec{v}_j} \frac{1}{k} |I_d^{j+1} - I_C^{j+1}|^k\end{aligned}$$

First, we will focus on the computation of $\nabla_{\vec{v}_j} \frac{1}{k} |I_d^j - I_C^j|^k$.

$$\begin{aligned}\nabla_{\vec{v}_j} \frac{1}{k} |I_d^j - I_C^j|^k &= (I_d^j - I_C^j) \nabla_{\vec{v}_j} (-I_C^j) \\ &= (I_C^j - I_d^j) \nabla_{\vec{v}_j} I_C^j \\ &= (I_C^j - I_d^j) \nabla_{\vec{v}_j} [\operatorname{arctan2}(\vec{v}_{j-1} - \vec{v_j}) - \operatorname{arctan2}(\vec{v}_{j+1} - \vec{v_j})]_{[0,2\pi)}.\end{aligned}$$

At this point, the previously mentioned simplifications come into play and we use $\arctan\left(\frac{v_{j-1}^y - v_j^y}{v_{j-1}^x - v_j^x}\right)$ instead of $\operatorname{arctan2}(\vec{v}_{j-1} - \vec{v_j})$ and neglect the modulo operator. In the next step, we need to compute the gradient

$$\nabla_{\vec{v}_j} \arctan\left(\frac{v_{j-1}^y - v_j^y}{v_{j-1}^x - v_j^x}\right).$$

Therefore, we define helper functions

$$f(\vec{v}) = \arctan\left(\frac{v^y}{v^x}\right)$$

and

$$g(\vec{v}_{j-1}, \vec{v}_j) = \begin{pmatrix} v_{j-1}^x - v_j^x \\ v_{j-1}^y - v_j^y \end{pmatrix}.$$

With these helper functions, we can write

$$\arctan\left(\frac{v_{j-1}^y - v_j^y}{v_{j-1}^x - v_j^x}\right) = (f \circ g)(\vec{v}_{j-1}, \vec{v}_j)$$

and use the two dimensional chain rule to stepwise compute the searched gradient.

$$\frac{\partial f(\vec{v})}{\partial v^x} = \frac{1}{1 + \left(\frac{v^y}{v^x}\right)^2} \left(-\frac{v^y}{(v^x)^2}\right) = -\frac{v^y}{(v^x)^2 + (v^y)^2}$$

$$\frac{\partial f(\vec{v})}{\partial v^y} = \frac{1}{1 + \left(\frac{v^y}{v^x}\right)^2} \frac{1}{v^x} = \frac{v^x}{(v^x)^2 + (v^y)^2}$$

$$\begin{aligned}\nabla_{v_j^x} g(\vec{v}_{j-1}, \vec{v}_j) &= (-1, 0)^T \\ \nabla_{v_j^y} g(\vec{v}_{j-1}, \vec{v}_j) &= (0, -1)^T\end{aligned}$$

With that, we can compute:

$$\begin{aligned}
\frac{\partial(f \circ g(\vec{v}_{j-1}, \vec{v}_j))}{\partial v_j^x} &= (\nabla f \circ g(\vec{v}_{j-1}, \vec{v}_j))^T \cdot \nabla_{v_j^x} g(\vec{v}_{j-1}, \vec{v}_j) \\
&= \begin{pmatrix} -\frac{v_{j-1}^y - v_j^y}{(v_{j-1}^x - v_j^x)^2 + (v_{j-1}^y - v_j^y)^2} \\ \frac{v_{j-1}^x - v_j^x}{(v_{j-1}^x - v_j^x)^2 + (v_{j-1}^y - v_j^y)^2} \end{pmatrix}^T \cdot \begin{pmatrix} -1 \\ 0 \end{pmatrix} \\
&= \frac{v_{j-1}^y - v_j^y}{(v_{j-1}^x - v_j^x)^2 + (v_{j-1}^y - v_j^y)^2} \\
&= \frac{v_{j-1}^y - v_j^y}{\|\vec{v}_{j-1} - \vec{v}_j\|_2^2}
\end{aligned}$$

And similarly:

$$\begin{aligned}
\frac{\partial(f \circ g(\vec{v}_{j-1}, \vec{v}_j))}{\partial v_j^y} &= (\nabla f \circ g(\vec{v}_{j-1}, \vec{v}_j))^T \cdot \nabla_{v_j^y} g(\vec{v}_{j-1}, \vec{v}_j) \\
&= \begin{pmatrix} -\frac{v_{j-1}^y - v_j^y}{(v_{j-1}^x - v_j^x)^2 + (v_{j-1}^y - v_j^y)^2} \\ \frac{v_{j-1}^x - v_j^x}{(v_{j-1}^x - v_j^x)^2 + (v_{j-1}^y - v_j^y)^2} \end{pmatrix}^T \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix} \\
&= -\frac{v_{j-1}^x - v_j^x}{(v_{j-1}^x - v_j^x)^2 + (v_{j-1}^y - v_j^y)^2} \\
&= \frac{v_j^x - v_{j-1}^x}{\|\vec{v}_{j-1} - \vec{v}_j\|_2^2}
\end{aligned}$$

Overall, we get

$$\nabla_{\vec{v}_j} \arctan\left(\frac{v_{j-1}^y - v_j^y}{v_{j-1}^x - v_j^x}\right) = \frac{1}{\|\vec{v}_{j-1} - \vec{v}_j\|_2^2} \begin{pmatrix} v_{j-1}^y - v_j^y \\ v_j^x - v_{j-1}^x \end{pmatrix}.$$

Thus, we can come back to:

$$\begin{aligned}
\nabla_{\vec{v}_j} \frac{1}{k} |I_d^j - I_C^j|^k &= (I_C^j - I_d^j) \nabla_{\vec{v}_j} \left(\arctan\left(\frac{v_{j-1}^y - v_j^y}{v_{j-1}^x - v_j^x}\right) - \arctan\left(\frac{v_{j+1}^y - v_j^y}{v_{j+1}^x - v_j^x}\right) \right) \\
&= (I_C^j - I_d^j) \left(\frac{1}{\|\vec{v}_{j-1} - \vec{v}_j\|_2^2} \begin{pmatrix} v_{j-1}^y - v_j^y \\ v_j^x - v_{j-1}^x \end{pmatrix} \right. \\
&\quad \left. - \frac{1}{\|\vec{v}_{j+1} - \vec{v}_j\|_2^2} \begin{pmatrix} v_{j+1}^y - v_j^y \\ v_j^x - v_{j+1}^x \end{pmatrix} \right).
\end{aligned}$$

For the neighboring vertices, we need:

$$\nabla_{\vec{v}_j} \arctan\left(\frac{v_j^y - v_{j-1}^y}{v_j^x - v_{j-1}^x}\right) \text{ and } \nabla_{\vec{v}_j} \arctan\left(\frac{v_j^y - v_{j+1}^y}{v_j^x - v_{j+1}^x}\right).$$

Therefore, we can use the same helper function

$$f(\vec{v}) = \arctan\left(\frac{v^y}{v^x}\right),$$

but we need different functions for g, as the arrangement of the vertex coordinates differs a bit. We introduce

$$g_-(\vec{v}_{j-1}, \vec{v}_j) = \begin{pmatrix} v_j^x - v_{j-1}^x \\ v_j^y - v_{j-1}^y \end{pmatrix} = -g(\vec{v}_{j-1}, \vec{v}_j).$$

and

$$g_+(\vec{v}_j, \vec{v}_{j+1}) = \begin{pmatrix} v_j^x - v_{j+1}^x \\ v_j^y - v_{j+1}^y \end{pmatrix}.$$

The gradients are

$$\nabla_{v_j^x} g_-(\vec{v}_{j-1}, \vec{v}_j) = (1,0)^T, \nabla_{v_j^y} g_-(\vec{v}_{j-1}, \vec{v}_j) = (0,1)^T,$$

$$\nabla_{v_j^x} g_+(\vec{v}_{j-1}, \vec{v}_j) = (1,0)^T, \nabla_{v_j^y} g_+(\vec{v}_{j-1}, \vec{v}_j) = (0,1)^T.$$

Thus, the Jacobian of both g_- and g_+ is the identity matrix and we can just neglect it in the following chain rules. For the previous vertex, we get

$$\begin{aligned}
\nabla_{\vec{v}_j} \frac{1}{k} |I_C^{j-1} - I_d^{j-1}|^k &= \operatorname{sgn}(I_C^{j-1} - I_d^{j-1}) |I_C^{j-1} - I_d^{j-1}|^{k-1} \cdot \\
&\qquad \nabla_{\vec{v}_j} \left(\arctan\left(\frac{v_{j-2}^y - v_{j-1}^y}{v_{j-2}^x - v_{j-1}^x}\right) - \arctan\left(\frac{v_j^y - v_{j-1}^y}{v_j^x - v_{j-1}^x}\right)\right) \\
&= \operatorname{sgn}(I_C^{j-1} - I_d^{j-1}) |I_C^{j-1} - I_d^{j-1}|^{k-1} \cdot \\
&\qquad \nabla_{\vec{v}_j} \left(- \arctan\left(\frac{v_j^y - v_{j-1}^y}{v_j^x - v_{j-1}^x}\right)\right) \\
&= \operatorname{sgn}(I_C^{j-1} - I_d^{j-1}) |I_C^{j-1} - I_d^{j-1}|^{k-1} \cdot \\
&\qquad (-\nabla_{\vec{v}_j} (f \circ g_-(\vec{v}_{j-1}, \vec{v}_{j-1}))) \\
&= \operatorname{sgn}(I_C^{j-1} - I_d^{j-1}) |I_C^{j-1} - I_d^{j-1}|^{k-1} \cdot \\
&\qquad (-(\nabla f) \circ g_-(\vec{v}_{j-1}, \vec{v}_{j-1})) \\
&= \operatorname{sgn}(I_C^{j-1} - I_d^{j-1}) |I_C^{j-1} - I_d^{j-1}|^{k-1} \cdot \\
&\qquad \left(-\frac{1}{\|\vec{v}_j - \vec{v}_{j-1}\|_2^2} (v_{j-1}^y - v_j^y, v_j^x - v_{j-1}^x)^T \right).
\end{aligned}$$

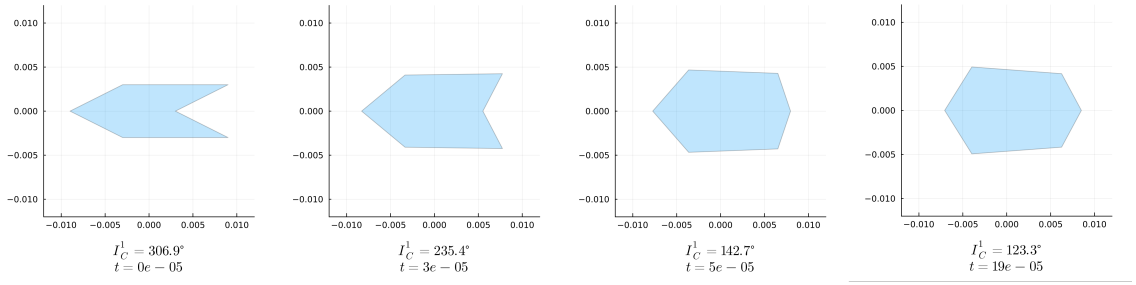

Figure 6: Click: *here*, to see the animation as a gif.

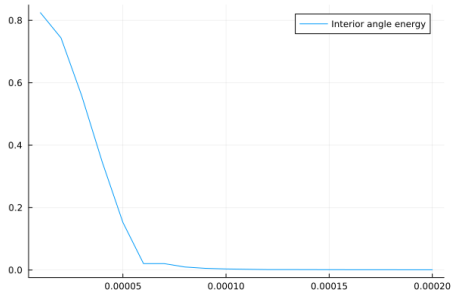

Figure 7

Finally, for the successor vertex, we get

$$\begin{aligned}
\nabla_{\vec{v}_j} \frac{1}{k} |I_C^{j+1} - I_d^{j+1}|^k &= \operatorname{sgn}(I_C^{j+1} - I_d^{j+1}) |I_C^{j+1} - I_d^{j+1}|^{k-1} \cdot \\
&\qquad \nabla_{\vec{v}_j} \left(\arctan\left(\frac{v_j^y - v_{j+1}^y}{v_j^x - v_{j+1}^x} \right) - \arctan\left(\frac{v_{j+2}^y - v_{j+1}^y}{v_{j+2}^x - v_{j+1}^x} \right) \right) \\
&= \operatorname{sgn}(I_C^{j+1} - I_d^{j+1}) |I_C^{j+1} - I_d^{j+1}|^{k-1} \cdot \\
&\qquad \nabla_{\vec{v}_j} \left(\arctan\left(\frac{v_j^y - v_{j+1}^y}{v_j^x - v_{j+1}^x} \right) \right) \\
&= \operatorname{sgn}(I_C^{j+1} - I_d^{j+1}) |I_C^{j+1} - I_d^{j+1}|^{k-1} \cdot \\
&\qquad \nabla_{\vec{v}_j} \left(f \circ g_+(\vec{v}_{j-1}, \vec{v}_{j-1}) \right) \\
&= \operatorname{sgn}(I_C^{j+1} - I_d^{j+1}) |I_C^{j+1} - I_d^{j+1}|^{k-1} \cdot \\
&\qquad (\nabla f) \circ g_+(\vec{v}_{j-1}, \vec{v}_{j-1}) \\
&= \operatorname{sgn}(I_C^{j+1} - I_d^{j+1}) |I_C^{j+1} - I_d^{j+1}|^{k-1} \cdot \\
&\qquad \left(\frac{1}{\|\vec{v}_j - \vec{v}_{j+1}\|_2^2} (v_{j+1}^y - v_j^y, v_j^x - v_{j+1}^x)^T \right).
\end{aligned}$$

□

Figure 6 illustrates the isolated effect of the interior angle force.

3.4 Overlap force

Unlike the previous energies, which act independently on each cell, the overlap force is the first to account for interactions between multiple cells, thereby introducing cell-to-cell interaction into the simulation.

Deforming overlap force

The first overlap force, that we want to introduce, is similar to the overlap force introduced in [Vog23]. It degenerates a cell overlap by influencing that cell shapes of the affected cells.
The challenging aspect of computing the overlap force lies in detecting overlaps within the cell system. An overlap is treated as a DF cell in its own right, composed of the vertices from each of the two overlapping cells that lie inside the other, along with the two intersection points where the cell boundaries intersect.
Once all overlaps have been identified, we apply a dynamic similar to that of the area force, but with a desired area of zero. This generates a force that acts to eliminate the overlap by reducing its area to zero. The resulting force is then applied to the vertices of the original cells that define the overlapping region.
The first step in detecting overlaps is identifying the intersection points between cell boundaries. Intersections can be identified by representing the cell edges as line segments and computing the intersection points between segments belonging to different cells.
Having found all intersections, we can apply the following algorithm, that can be used to compute all overlaps between two cells.

Algorithm 3.9. ***Computation of a discrete overlap***

INPUT:

- *Discrete cells C and ζ*
- *List I of unused intersections of C and ζ*

```
function constructOverlap(C, ζ, I)
    usedIntersections = List{Intersection}(I[1])
    newOverlap = List{Vertices}(I[1])
    currentIntersection = I[1]
    for counter = 1 : length(I) do
        if counter is even then
            newPath, newIntersection = findPath(currentIntersection, C, I)
        else
            newPath, newIntersection = findPath(currentIntersection, ζ, I)
        end if
        append!(newOverlap, newPath)
        if newIntersection == I[1] then
            return newOverlap, usedIntersections
        else
            append!(newOverlap, newIntersection)
```

```
            append!(usedIntersections, newIntersection)
            currentIntersection = newIntersection
        end if
    end for
end function
```

OUTPUT:

- *A single intersection 'newOverlap' which occurs between C and ζ and which uses vertices from C and ζ as well as only intersections from I*
- *A list 'usedIntersections' of all intersection that are used in 'newOverlap'*

The algorithm begins by selecting the first intersection point $I[1]$ from the list I as the initial vertex of the overlap cell 'newOverlap'. This point is also added to the list 'usedIntersections'
Next, the function 'getOverlap' calls another function, 'findPath', which determines the path along the discrete cell ζ from the current intersection point to the next intersection in I encountered while traversing the edges of ζ. This next intersection is also returned by the function. The identified path is a list of vertices in ζ that lie strictly between the two intersections. It may be empty if the next intersection occurs on the same edge as the current one. Both the path and the newly found intersection are appended to 'newOverlap', and the intersection is also added to the list usedIntersections.
Since each intersection implies changing the cell from which the overlapping cell uses the edges, 'findPath' is now applied to the other cell. Again, it will deliver the next intersection as well as a list of the in between laying vertices. The vertex list always gets appended to 'newOverlap'.
If the newly found intersection is equal to the initial intersection $I[1]$, then the construction of the discrete overlap cell 'newOverlap' is complete. At this point, both 'newOverlap' and 'usedIntersections' can be returned by the function 'constructOverlap'.
Otherwise, the newly found intersection is appended to both 'newOverlap' and 'usedIntersections', and the process continues by calling 'findPath' on the other discrete cell. This step is repeated until the starting intersection is reached, completing the overlap cell construction.
Once an overlap between C and ζ has been successfully extracted, all intersections used in its construction can be removed from the list I, since each intersection point belongs to exactly one overlap. As long as I is not empty, the function 'constructOverlap' can be called again with the updated list to extract the next overlap. When I is empty, we can be certain that all intersections between C and ζ have been processed, and thus all overlaps between the two cells have been identified.

Each time 'findPath' is called, it is not immediately clear in which direction the function should traverse the vertices of the given cell. However, the correct direction can be determined using the following approach.
Starting from the current intersection passed into the function, move a small distance in one direction along the edge of the given cell where the intersection is

located. Next, check whether this new point lies within the boundaries of the other cell as well. If the point is found in both cells, the chosen direction is correct. If not, then the opposite direction must be used.
A simple method to determine whether a point lies inside a polygon is to draw a ray from the point to the outside of the polygon. The number of intersections between the ray and the polygon's edges determines the point's position. If the number of intersections is odd, the point is inside the polygon. If it is even, the point is outside the polygon.

After introducing the method for detecting overlaps, we can now define the overlap force, which acts on the cell vertices involved in an overlap. This force is first computed based on the geometry of the overlap and then distributed to the corresponding vertices of the original cells.

Definition 3.10. Overlap energy
Let C_i and C_m be two cells from the system $\vec{C}$ and $\Omega_{i,m}$ be the set of all overlaps that appear between C_i and C_m, like explained above. Then, the total overlap energy $O_k : (\mathbb{R}^{2N_V})^{N_C} \to \mathbb{R}$ of the cell system is given by the formula

$$O_k(\vec{C}) = \sum_{i=1}^{N_C} \left(\sum_{m=i+1}^{N_C} \left(\sum_{D \in \Omega_{i,m}} \frac{1}{k} |A_D|^k \right) \right), \tag{7}$$

where A_D is the area of the overlap D.
We define the inner bracket to be the overlap energy of the cell pair C_i and C_m

$$O_k^{i,m} : (\mathbb{R}^{2N_V})^2 \to \mathbb{R}$$
$$O_k^{i,m}(C_i, C_m) = \sum_{D \in \Omega_{i,m}} \tfrac{1}{k} |A_D|^k.$$

To decrease the overlap areas during the simulation, we evaluate the gradient flow of the area energy with a desired area of zero which indicates the direction of motion for each vertex for reducing the overlap areas.

Definition 3.11. Intersection point and adjacent vertices
The vertices of an overlap cell D can be divided into the vertices that are either in C_i in C_m, we call that set

$$V(D) = \{\vec{v} \in D \,|\vec{v} \in C_i \cup C_m\},$$

and into the vertices that are neither in C_i nor C_m, named

$$W(D) = D \backslash V(D).$$

All overlap vertices in $W(D)$ are intersections between the cells C_i and C_m.
Each intersection $\vec{w}$ is dependent on two vertices of each cell that limit the edges that intersect, where the intersection point $\vec{w}$ arises.
We call those four vertices **adjacent** to the intersection $\vec{w}$. All intersection adjacent vertices will get an extra deforming overlap dynamic applied, since they influence the overlap area, and thus the overlap energy, via the intersection point they create.

From each cell we get one vertex that is part of the overlap cell, called the inside vertex, and one vertex that is not part of the overlap, called the outside vertex. For each intersection $\vec{w} \in W(D)$, we call its adjacent vertices

$$\text{adj}(\vec{w}) = \{\vec{v}^i_{\text{in}}, \vec{v}^i_{\text{out}}, \vec{v}^m_{\text{in}}, \vec{v}^m_{\text{out}}\}.$$

Given the four adjacent vertices, we can always compute the intersection with the function:

$$w : (\mathbb{R}^2)^4 \to \mathbb{R}^2,$$

$$w(\vec{v}^i_{\text{out}}, \vec{v}^i_{\text{in}}, \vec{v}^m_{\text{out}}, \vec{v}^m_{\text{in}}) = \vec{v}^i_{\text{out}} + \frac{(\vec{v}^m_{\text{out}} - \vec{v}^i_{\text{out}}) \times (\vec{v}^m_{\text{in}} - \vec{v}^m_{\text{out}})}{(\vec{v}^i_{\text{in}} - \vec{v}^i_{\text{out}}) \times (\vec{v}^m_{\text{in}} - \vec{v}^m_{\text{out}})}(\vec{v}^i_{\text{in}} - \vec{v}^i_{\text{out}}),$$

where $(a^x, a^y)^T \times (b^x, b^y)^T = a^x b^y - a^y b^x$ denotes the two dimensional cross product.

Proposition 3.12. ***Partial derivatives of intersection points***
We use $w(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in})$ to compute the influence of the adjacent vertices to the intersection point via their partial derivatives. In this proposition, we compute the needed partial derivatives.
We define the following helper functions:

$$f : (\mathbb{R}^2)^3 \to \mathbb{R} \quad f(\vec{v}^i_{out}, \vec{v}^m_{out}, \vec{v}^m_{in}) = (\vec{v}^m_{out} - \vec{v}^i_{out}) \times (\vec{v}^m_{in} - \vec{v}^m_{out})$$

$$g : (\mathbb{R}^2)^4 \to \mathbb{R} \quad g(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in}) = (\vec{v}^i_{in} - \vec{v}^i_{out}) \times (\vec{v}^m_{in} - \vec{v}^m_{out})$$

$$t : (\mathbb{R}^2)^4 \to \mathbb{R} \quad t(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in}) = \frac{f(\vec{v}^i_{out}, \vec{v}^m_{out}, \vec{v}^m_{in})}{g(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in})}$$

$$w : (\mathbb{R}^2)^4 \to \mathbb{R}^2 \quad w(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in}) = \vec{v}^i_{out} + t(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in})(\vec{v}^i_{in} - \vec{v}^i_{out})$$

It is sufficient to just compute the partial derivatives with respect to $\vec{v}^i_{in}$ and $\vec{v}^i_{out}$. If we want the dynamic for the vertices of the other cell, we just switch the arrangement of the arguments (switch $\vec{v}^i_{in}$ with $\vec{v}^j_{in}$ and $\vec{v}^i_{out}$ with $\vec{v}^j_{out}$) and then use the same partial derivatives as for the first cell.
The partial derivatives are:

$$\frac{\partial w(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in})}{\partial \vec{v}^i_{out}} : (\mathbb{R}^2)^4 \to \mathbb{R}^{2\times 2},$$

(8)
$$\frac{\partial w(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in})}{\partial \vec{v}^i_{out}} = (1-t)I_2 + \frac{g-f}{g^2}(\vec{v}^i_{in} - \vec{v}^i_{out}) \begin{pmatrix} -(v^{m,y}_{in} - v^{m,y}_{out}) \\ v^{m,x}_{in} - v^{m,x}_{out} \end{pmatrix}^T,$$

$$\frac{\partial w(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in})}{\partial \vec{v}^i_{in}} : (\mathbb{R}^2)^4 \to \mathbb{R}^{2\times 2},$$

(9)
$$\frac{\partial w(\vec{v}^i_{out}, \vec{v}^i_{in}, \vec{v}^m_{out}, \vec{v}^m_{in})}{\partial \vec{v}^i_{in}} = tI_2 \; + \; \frac{f}{g^2}(\vec{v}^i_{in} - \vec{v}^i_{out}) \begin{pmatrix} -(v^{m,y}_{in} - v^{m,y}_{out}) \\ v^{m,x}_{in} - v^{m,x}_{out} \end{pmatrix}^T.$$

Proof.

For the ease of notation, we will drop the arguments, but keep in mind that f, g and t are dependent on the vertices.
First of all, we compute the gradients of f and g:

$$\begin{aligned}\nabla_{\vec{v}^{\,i}_{out}} f &= \nabla_{\vec{v}^{\,i}_{out}}[(v^{m,x}_{out} - v^{i,x}_{out})(v^{m,y}_{in} - v^{m,y}_{out}) - (v^{m,y}_{out} - v^{i,y}_{out})(v^{m,x}_{in} - v^{m,x}_{out})] \\ &= \begin{pmatrix} -(v^{m,y}_{in} - v^{m,y}_{out}) \\ v^{m,x}_{in} - v^{m,x}_{out} \end{pmatrix},\end{aligned}$$

$$\begin{aligned}\nabla_{\vec{v}^{\,i}_{in}} f &= \nabla_{\vec{v}^{\,i}_{in}}[(v^{m,x}_{out} - v^{i,x}_{out})(v^{m,y}_{in} - v^{m,y}_{out}) - (v^{m,y}_{out} - v^{i,y}_{out})(v^{m,x}_{in} - v^{m,x}_{out})] \\ &= \begin{pmatrix} 0 \\ 0 \end{pmatrix},\end{aligned}$$

$$\begin{aligned}\nabla_{\vec{v}^{\,i}_{out}} g &= \nabla_{\vec{v}^{\,i}_{out}}[(v^{i,x}_{in} - v^{i,x}_{out})(v^{m,y}_{in} - v^{m,y}_{out}) - (v^{i,y}_{in} - v^{i,y}_{out})(v^{m,x}_{in} - v^{m,x}_{out})] \\ &= \begin{pmatrix} -(v^{m,y}_{in} - v^{m,y}_{out}) \\ v^{m,x}_{in} - v^{m,x}_{out} \end{pmatrix},\end{aligned}$$

$$\begin{aligned}\nabla_{\vec{v}^{\,i}_{in}} g &= \nabla_{\vec{v}^{\,i}_{in}}[(v^{i,x}_{in} - v^{i,x}_{out})(v^{m,y}_{in} - v^{m,y}_{out}) - (v^{i,y}_{in} - v^{i,y}_{out})(v^{m,x}_{in} - v^{m,x}_{out})] \\ &= \begin{pmatrix} v^{m,y}_{in} - v^{m,y}_{out} \\ -(v^{m,x}_{in} - v^{m,x}_{out}) \end{pmatrix}.\end{aligned}$$

Now, we can succeed with the gradients of t:

$$\begin{aligned}\nabla_{\vec{v}^{\,i}_{out}} t &= \nabla_{\vec{v}^{\,i}_{out}} \frac{f}{g} \\ &= \frac{(\nabla_{\vec{v}^{\,i}_{out}} f)g - (\nabla_{\vec{v}^{\,i}_{out}} g)f}{g^2} \\ &= \frac{1}{g^2}\left(\begin{pmatrix} -(v^{m,y}_{in} - v^{m,y}_{out}) \\ v^{m,x}_{in} - v^{m,x}_{out} \end{pmatrix} g - \begin{pmatrix} -(v^{m,y}_{in} - v^{m,y}_{out}) \\ v^{m,x}_{in} - v^{m,x}_{out} \end{pmatrix} f\right) \\ &= \frac{g - f}{g^2}\begin{pmatrix} -(v^{m,y}_{in} - v^{m,y}_{out}) \\ v^{m,x}_{in} - v^{m,x}_{out} \end{pmatrix}\end{aligned}$$

$$
\begin{aligned}
\nabla_{\vec{v}_{in}^{\,i}} t &= \nabla_{\vec{v}_{in}^{\,i}} \frac{f}{g} \\
&= \frac{(\nabla_{\vec{v}_{in}^{\,i}} f)g - (\nabla_{\vec{v}_{in}^{\,i}} g)f}{g^2} \\
&= \frac{1}{g^2}\left(- \begin{pmatrix} v_{in}^{m,y} - v_{out}^{m,y} \\ -(v_{in}^{m,x} - v_{out}^{m,x}) \end{pmatrix} f \right). \\
&= \frac{f}{g^2} \begin{pmatrix} -(v_{in}^{m,y} - v_{out}^{m,y}) \\ v_{in}^{m,x} - v_{out}^{m,x} \end{pmatrix}.
\end{aligned}
$$

And finally, we can compute the partial derivatives of $w = (w_1, w_2)^T$*:*

$$
\begin{aligned}
&\frac{\partial w_1}{\partial v_{out}^{i,x}} = 1 + \frac{\partial t}{\partial v_{out}^{i,x}}(v_{in}^{i,x} - v_{out}^{i,x}) - t, \quad \frac{\partial w_1}{\partial v_{out}^{i,y}} = 0 + \frac{\partial t}{\partial v_{out}^{i,y}}(v_{in}^{i,x} - v_{out}^{i,x}) + 0, \\
&\frac{\partial w_2}{\partial v_{out}^{i,x}} = 0 + \frac{\partial t}{\partial v_{out}^{i,x}}(v_{in}^{i,y} - v_{out}^{i,y}) + 0, \quad \frac{\partial w_2}{\partial v_{out}^{i,y}} = 1 + \frac{\partial t}{\partial v_{out}^{i,y}}(v_{in}^{i,y} - v_{out}^{i,y}) - t, \\
&\Rightarrow \frac{\partial w}{\vec{v}_{out}^{\,i}} = (1-t)I_2 + (\vec{v}_{in}^{\,i} - \vec{v}_{out}^{\,i})(\nabla_{\vec{v}_{out}^{\,i}} t)^T \\
&\qquad = (1-t)I_2 + (\vec{v}_{in}^{\,i} - \vec{v}_{out}^{\,i}) \left(\frac{g-f}{g^2} \begin{pmatrix} -(v_{in}^{m,y} - v_{out}^{m,y}) \\ v_{in}^{m,x} - v_{out}^{m,x} \end{pmatrix} \right)^T \\
&\qquad = (1-t)I_2 + \frac{g-f}{g^2}(\vec{v}_{in}^{\,i} - \vec{v}_{out}^{\,i}) \begin{pmatrix} -(v_{in}^{m,y} - v_{out}^{m,y}) \\ v_{in}^{m,x} - v_{out}^{m,x} \end{pmatrix}^T,
\end{aligned}
$$

$$
\begin{aligned}
&\frac{\partial w_1}{\partial v_{in}^{i,x}} = \frac{\partial t}{\partial v_{in}^{i,x}}(v_{in}^{i,x} - v_{out}^{i,x}) + t, \quad \frac{\partial w_1}{\partial v_{in}^{i,y}} = \frac{\partial t}{\partial v_{in}^{i,y}}(v_{in}^{i,x} - v_{out}^{i,x}) + 0, \\
&\frac{\partial w_2}{\partial v_{in}^{i,x}} = \frac{\partial t}{\partial v_{in}^{i,x}}(v_{in}^{i,y} - v_{out}^{i,y}) + 0, \quad \frac{\partial w_2}{\partial v_{in}^{i,y}} = \frac{\partial t}{\partial v_{in}^{i,y}}(v_{in}^{i,y} - v_{out}^{i,y}) + t,
\end{aligned}
$$

$$\Rightarrow \frac{\partial w}{\vec{v}_{in}^{\,i}} = tI_2 + (\vec{v}_{in}^{\,i} - \vec{v}_{out}^{\,i})(\nabla_{\vec{v}_{in}^{\,i}} t)^T$$

$$= tI_2 + (\vec{v}_{in}^{\,i} - \vec{v}_{out}^{\,i}) \left(\frac{f}{g^2} \begin{pmatrix} -(v_{in}^{m,y} - v_{out}^{m,y}) \\ v_{in}^{m,x} - v_{out}^{m,x} \end{pmatrix} \right)^T$$

$$= \; tI_2 \; + \; \frac{f}{g^2} (\vec{v}_{in}^{\,i} - \vec{v}_{out}^{\,i}) \begin{pmatrix} -(v_{in}^{m,y} - v_{out}^{m,y}) \\ v_{in}^{m,x} - v_{out}^{m,x} \end{pmatrix}^T .$$

Proposition 3.13. ***Deforming overlap force***
Let $\Omega_{i,m}$ be the set of all overlaps between the cells C_i and C_m. Each overlap cell $D \in \Omega_{i,m}$ is a list of vertices, that form the overlap, just like a DF cell.
The deforming overlap gradient is then given by

$$\nabla_{\vec{v}_j^{\,i}} O_k(\vec{C}) = \sum_{D \in \Omega_{i,m}} |A_D|^{k-1} \Bigg(\mathbf{1}_{V(D)}(\vec{v}_j^{\,i}) \nabla_{\vec{v}_j^{\,i}} A(D) + \tag{10}$$
$$+ \sum_{\vec{w} \in W(D)} \left(\mathbf{1}_{out(w)}(\vec{v}_j^{\,i}) \frac{\partial \vec{w}}{\partial \vec{v}_{j,out}^{\,i}} + \mathbf{1}_{in(w)}(\vec{v}_j^{\,i}) \frac{\partial \vec{w}}{\partial \vec{v}_{j,in}^{\,i}} \right) \nabla_{\vec{w}} A(D) \Bigg),$$

for all $1 \leqslant j \leqslant N_V$ and $1 \leqslant i \leqslant N_C$, where $out(w)$, $in(w) \subset adj(w)$ denote the sets of outside and inside overlap-adjacent vertices, respectively.
Note, that the Formulas 8 and 9 define $\frac{\partial \vec{w}}{\partial \vec{v}_{j,out}^{\,i}}$ and $\frac{\partial \vec{w}}{\partial \vec{v}_{j,in}^{\,i}}$, respectively.
The difference between $\nabla_{\vec{v}_j^{\,i}} A(D)$ and $\nabla_{\vec{w}} A(D)$ is that $\nabla_{\vec{v}_j^{\,i}} A(D)$ uses the neighboring overlap vertices of $\vec{v}_j^{\,i}$ itself in the Area Gradient Formula 2, whereas $\nabla_{\vec{w}} A(D)$ uses the neighbors of the corresponding intersection point in D, which is not the same overlap vertex as $\vec{v}_j^{\,i}$. A_D is the area of the overlap D.
The indicator function $\mathbf{1}_A(\vec{v})$ equals one, if $\vec{v} \in A$ and is zero otherwise.
The deforming overlap force $F_k^{(\hat{O})} : (\mathbb{R}^{2N_V})^{N_C} \to (\mathbb{R}^{2N_V})^{N_C}$ that gets applied on the whole cell system $\vec{C} = (C_1, \ldots, C_{N_C})$ is then given by

$$F_k^{(\hat{O})}(\vec{C}) = -(\nabla_{\vec{v}_1^{\,1}} O_k(\vec{C}), \ldots, \nabla_{\vec{v}_{N_V}^{\,1}} O_k(\vec{C}), \; \cdots, \vec{v}_1^{\,N_C} O_k(\vec{C}), \ldots, \nabla_{\vec{v}_{N_V}^{\,N_C}} O_k(\vec{C}))^T.$$

For addressing the overlap force that acts on cell $1 \leqslant i \leqslant N_C$, we define

$$F_{k,i}^{(\hat{O})} : (\mathbb{R}^{2N_V})^{N_C} \to (\mathbb{R}^{2N_V}),$$
$$F_{k,i}^{(\hat{O})}(\vec{C}) = -(\nabla_{\vec{v}_1^{\,i}} O_k(\vec{C}), \ldots, \nabla_{\vec{v}_{N_V}^{\,i}} O_k(\vec{C}))^T.$$

Proof.
Although we did not noted it like this before, we must be aware that A_D is actually dependent on all overlap vertices, e.g. $A_D = A(D)$. We will also use that notation in the coming computation. Since the area $A(D)$ is always positive, we can drop the

absolute value.
We aim to compute

$$\begin{aligned}\nabla_{\vec{v}_j^i} O_k(\vec{C}) &= \nabla_{\vec{v}_j^i} \sum_{i=1}^{N_C} \left(\sum_{m=i+1}^{N_C} \left(\sum_{D \in \Omega_{i,m}} \frac{1}{k} A_D^k \right) \right) \\ &= \nabla_{\vec{v}_j^i} \sum_{m \neq i} O_k^{i,m}(C_i, C_m) \\ &= \nabla_{\vec{v}_j^i} \sum_{m \neq i} \sum_{D \in \Omega_{i,m}} \frac{1}{k} A(D)^k \\ &= \sum_{m \neq i} \sum_{D \in \Omega_{i,m}} \nabla_{\vec{v}_j^i} \frac{1}{k} A(D)^k.\end{aligned}$$

Now, there are different cases.
Case 1: $\vec{v}_j^i \notin D$ *and* $\vec{v}_j^i \notin adj(\vec{w})\ \forall \vec{w} \in W(D)$
In the first case, the considered vertex is neither a vertex from the overlap cell D*, nor adjacent to any intersection point.*
Hence, this vertex has zero impact on the overlap and its gradient is zero:

$$\nabla_{\vec{v}_j^i} \frac{1}{k} A_D^k = 0.$$

Case 2: $\vec{v}_j^i \notin D$ *and* $\exists \vec{w} \in W(D) :\ \vec{v}_j^i \in adj(\vec{w})$
For case 2, we consider a vertex that is not directly an overlap vertex, but it influences the overlap cell by influencing an intersection point $\vec{w}$*. This means that* $\vec{v}_j^i$ *is an outside adjacent vertex of the intersection* $\vec{w}$*. We compute:*

$$\begin{aligned}\nabla_{\vec{v}_j^i} \frac{1}{k} A(D)^k &= A_D^{k-1} \nabla_{\vec{v}_j^i} A(D) \\ &= A_D^{k-1} \sum_{\vec{w} \in W(D)} \mathbf{1}_{out(w)}(\vec{v}_j^i) \frac{\partial \vec{w}}{\partial \vec{v}_{j,out}^i} \nabla_{\vec{w}} A(D),\end{aligned}$$

where, according to Equation 8

$$\frac{\partial \vec{w}}{\partial \vec{v}_{j,out}^i} = (1-t) I_2 + \frac{g-f}{g^2} (\vec{v}_{in}^i - \vec{v}_j^i) \begin{pmatrix} -(v_{in}^{i,y} - v_j^{i,y}) \\ v_{in}^{i,x} - v_j^{i,x} \end{pmatrix}^T,$$

because $\vec{v}_j^i$ *is an outside vertex in this case and* $\vec{v}_{in}^i$ *is the vertex adjacent to* $\vec{v}_j^i$ *in cell* i*, such that these vertices build the edge causing the intersection.*
The gradient $\nabla_{\vec{w}} A(D)$ *can easily be computed via the Shoelace Formula 3.1, as in the area gradient from Formula 2:*

$$\nabla_{\vec{w}} A(D) = \frac{1}{2} \begin{pmatrix} d_{j+1}^y - d_{j-1}^y \\ d_{j-1}^x - d_{j+1}^x \end{pmatrix},$$

where $\vec{d}^D_{j-1} = (d^x_{j-1}, d^y_{j-1})^T$ *and* $\vec{d}^D_{j+1} = (d^x_{j+1}, d^y_{j+1})^T$ *are the neighboring vertices of the intersection* $\vec{w}$ *in the overlap* D*.*

Case 3: $\vec{v}^i_j \in D$ *and* $\vec{v}^i_j \notin adj(\vec{w})\ \forall \vec{w} \in W(D)$
Now, $\vec{v}^i_j$ *is a pure inside overlap vertex, in the sence that it does not have an intersection point as a neighbor in the overlap. This dynamic is quite easy, since we just have to use the Shoelace Formular 3.1 for once more:*

$$\begin{aligned}\nabla_{\vec{v}^i_j} \frac{1}{k} A(D)^k &= A_D^{k-1} \nabla_{\vec{v}^i_j} A(D) \\ &= A_D^{k-1} \mathbf{1}_{V(D)}(\vec{v}^i_j) \frac{1}{2} \begin{pmatrix} d^y_{j+1} - d^y_{j-1} \\ d^x_{j-1} - d^x_{j+1} \end{pmatrix},\end{aligned}$$

where $\vec{d}^D_{j-1} = (d^x_{j-1}, d^y_{j-1})^T$ *and* $\vec{d}^D_{j+1} = (d^x_{j+1}, d^y_{j+1})^T$ *are the neighboring vertices of* $\vec{v}^i_j$ *in the overlap* D*.*

Case 4: $\vec{v}^i_j \in D$ *and* $\exists \vec{w} \in W(D) :\ \vec{v}^i_j \in adj(\vec{w})$
In the last case, the considered vertex is an overlap vertex and also adjacent to at least one intersection point. Thus, we have to add both dynamics from the last two cases and then use the partial derivative for inside vertices.

$$\begin{aligned}\nabla_{\vec{v}^i_j} \frac{1}{k} A(D)^k &= A_D^{k-1} \nabla_{\vec{v}^i_j} A(D) \\ &= A_D^{k-1} \left(\mathbf{1}_{V(D)}(\vec{v}^i_j) \nabla_{\vec{v}^i_j} A(D) + \sum_{\vec{w} \in W(D)} \mathbf{1}_{adj(w)}(\vec{v}^i_j) \frac{\partial \vec{w}}{\partial \vec{v}^i_j} \nabla_{\vec{w}} A(D) \right) \\ &= A_D^{k-1} \Bigg(\mathbf{1}_{V(D)}(\vec{v}^i_j) \nabla_{\vec{v}^i_j} A(D) + \\ &\quad + \sum_{\vec{w} \in W(D)} \left(\mathbf{1}_{out(w)}(\vec{v}^i_j) \frac{\partial \vec{w}}{\partial \vec{v}^i_{j,out}} \nabla_{\vec{w}} A(D) + \mathbf{1}_{in(w)}(\vec{v}^i_j) \frac{\partial \vec{w}}{\partial \vec{v}^i_{j,in}} \nabla_{\vec{w}} A(D) \right) \Bigg) \\ &= A_D^{k-1} \Bigg(\mathbf{1}_{V(D)}(\vec{v}^i_j) \nabla_{\vec{v}^i_j} A(D) + \\ &\quad + \sum_{\vec{w} \in W(D)} \left(\mathbf{1}_{out(w)}(\vec{v}^i_j) \frac{\partial \vec{w}}{\partial \vec{v}^i_{j,out}} + \mathbf{1}_{in(w)}(\vec{v}^i_j) \frac{\partial \vec{w}}{\partial \vec{v}^i_{j,in}} \right) \nabla_{\vec{w}} A(D) \Bigg),\end{aligned}$$

where $out(w)$, $in(w) \subset adj(w)$ *denote the sets of outside and inside overlap-adjacent vertices, respectively.*
The difference between $\nabla_{\vec{v}^i_j} A(D)$ *and* $\nabla_{\vec{w}} A(D)$ *is, that* $\nabla_{\vec{v}^i_j} A(D)$ *uses the neighboring overlap vertices of* $\vec{v}^i_j$ *itself in the Area Gradient Formula 2, whereas* $\nabla_{\vec{w}} A(D)$ *uses the neighbors of the according intersection point in* D *which is not the same overlap vertex as* $\vec{v}^i_j$*.*
Overall:
Actually, Case 4 already provides the final formulation, since in the other cases the additional terms vanish due to the indicator functions.

□

Figure 16 illustrates the interaction between two overlapping cells, highlighting the effect of the overlap force on their vertices.

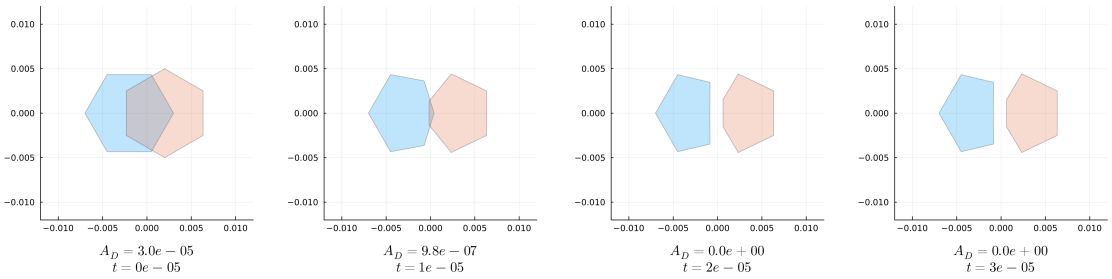

Figure 8: Click: *here*, to see the animation as a gif.

Bounce overlap force

While the previously introduced overlap force effectively reduces cell overlap by deforming the cells' shapes, it does not directly separate them spatially—leaving cells temporarily stuck together, relying on random Brownian motion to diffuse apart. With just that force, it is hard to compare the DF cell model to the hard sphere cell model, where overlaps are solved by reflecting them away from each other, resulting in a real distance that both cells have after a really small amount of time.
To address this limitation and ensure a smoother conceptual and mechanical transition from the hard disc cell model, where non deformable cells simply bounce off one another, we introduce a second overlap degeneration force. This additional force, which we refer to as the bounce overlap force, acts not by changing cell shape but by actively transporting overlapping cells away from each other. This mechanism captures the spatial repulsion characteristic of rigid body interactions while complementing the shape based degeneration of overlaps in deformable cells. In the end, we will use a combination of both overlap forces to get a nice transition from the HCSM to the DF cell model.
In [BC12] overlapping cells with a radius of r that have a centre-to-centre distance of $2r - a$ will be reflective apart in one time step (that is 10^{-5}), resulting in a distance

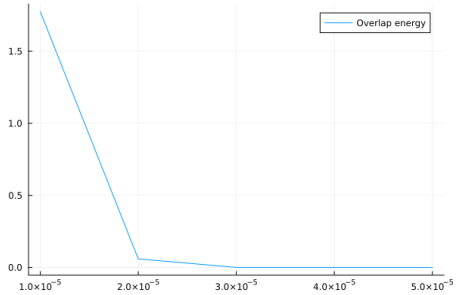

Figure 9

of $2r + a$ between the two cell centres afterwards.
The following force does the exact same for our DF cells. But we need the following assumptions:

1. Our DF cells model circular discs.
2. The cells have a radius of $r \in \mathbb{R}_{>0}$.
3. We can compute the cell centre $\vec{x} = \frac{1}{N_V} \sum_{j=1}^{N_v} \vec{v}_j$ which will be used to determine the distance between two cells.

Definition 3.14. Bounce overlap force

Let us consider two DF cells C_i and C_l, with centres at $\vec{x}_i$ and $\vec{x}_l$, respectively. We assume that each cell has a fixed radius $r > 0$. We define the vector $\mathrm{d}\bar{o}_{i,l} \in \mathbb{R}^2$, which is applied equally to all vertices of cell C_i, representing the repulsive overlap force caused by cell C_l. It is given by

$$\mathrm{d}\bar{o}_{i,l} = \mathbf{1}_{\|\vec{x}_i - \vec{x}_l\|_2 < 2r}(2r - \|\vec{x}_i - \vec{x}_l\|_2)\frac{\vec{x}_i - \vec{x}_l}{\|\vec{x}_i - \vec{x}_l\|_2}.$$

This force is zero if the distance between the cell centres satisfies $\|\vec{x}_i - \vec{x}_l\|_2 \geqslant 2r$. Otherwise, if the cells overlap (i.e., $\|\vec{x}_i - \vec{x}_l\|_2 < 2r$), the vector $\mathrm{d}\bar{o}_{i,l}$ points from $\vec{x}_l$ to $\vec{x}_i$ and has magnitude equal to the overlap depth $a = 2r - \|\vec{x}_i - \vec{x}_l\|_2$. At the same time in the simulation, the same magnitude of displacement is applied to all vertices of C_l in the opposite direction, i.e., along $\vec{x}_l - \vec{x}_i$. This means that all vertices of cell C_i are displaced away from C_l in the direction $\vec{x}_i - \vec{x}_l$ such that the resulting displacement is sufficient to separate the two cells' centres by exactly $2r + a$.
The force $F_{i,l}^{(\bar{O})}$ that acts on cell C_i due to its interaction with cell C_l is given by

$$F_{i,l}^{(\bar{O})}(C_i, C_l) = (\mathrm{d}\bar{o}_{i,l}, \ldots, \mathrm{d}\bar{o}_{i,l})^T \in \mathbb{R}^{2N_V},$$

where the vector $\mathrm{d}\bar{o}_{i,l}$ is repeated N_V times, once for each vertex of cell C_i.
The total bounce overlap force $F_i^{(\bar{O})} : (\mathbb{R}^{2N_V})^{N_C} \to \mathbb{R}^{2N_V}$ acting on cell C_i due to all other cells in the system is then defined as

$$F_i^{(\bar{O})}(\vec{C}) = \sum_{l \neq i} F_{i,l}^{(\bar{O})}(C_i, C_l).$$

In order to achieve a similarly fast degeneration of the overlap as in [BC12], we need to scale the force with the scaling factor $\alpha^{(\bar{O})} = 10^5$ as the time needed to resolve such an overlap in [BC12] was always one time step which is 10^{-5}.
To account for varying cell stiffness, we introduce a new parameter $h \in [0, 1]$ that controls how 'hard' the cells are. The total overlap force is then defined as a weighted combination of two overlap force types:

$$F^{(\mathbf{O})} = h \cdot F^{(\bar{O})} + (1 - h) \cdot F^{(\hat{O})},$$

where $F^{(\bar{O})}$ denotes the bounce-off overlap force and $F^{(\hat{O})}$ the shape-deforming overlap force.
When $h = 1$, the cells are maximally stiff. In this case, shape deformation is entirely suppressed: all overlaps are resolved solely through the bounce off mechanism, and the shape preserving forces become redundant since the cells always retain their desired configurations.
As h decreases, the cells become progressively softer. The influence of the bounce-off force diminishes, while the shape-deforming overlap force gains dominance, allowing cells to deform more in response to contact with neighbors.
With the introduction of the hardness parameter h, we have established a mechanism that enables a smooth transition from the HSCM dynamics introduced in [BC12] to our new DF cell model.
For $h = 1$, the dynamics are identical to the original HSCM model, as we will show in the following chapter. By gradually decreasing h, we can continuously adapt the system behavior toward the pure deformable (DF) cell model. In this way, we can systematically investigate how the dynamics evolve between the two regimes. In the limiting case $h = 0$, we recover the DF dynamics without the bounce overlap force term.

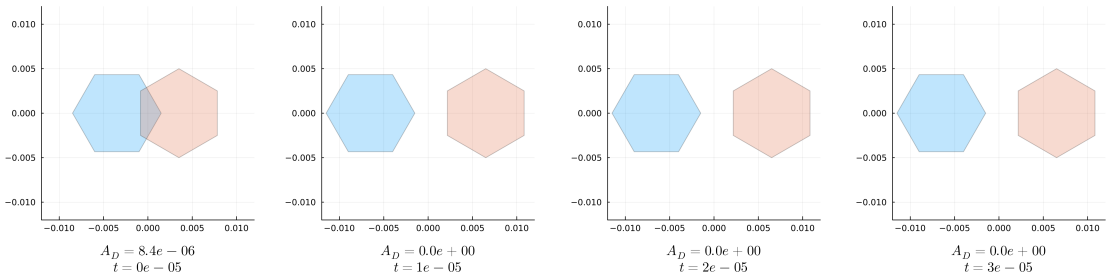

Figure 10: Click: *here*, to see the animation as a gif.

3.5 The DF SDE

Having defined all individual force contributions acting on the cell vertices, we now combine them to formulate the full dynamics of the system in terms of a stochastic

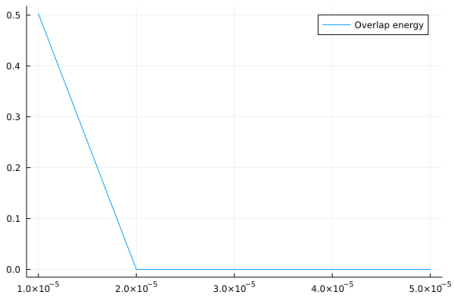

Figure 11

differential equation.
One challenging aspect that remains is the appropriate scaling of all forces. It has become evident that the system is highly sensitive to these scaling parameters. If the shape-recovering forces are too small, the recovery process is excessively slow. Conversely, if they are too large, the numerical integration scheme tends to become unstable.
To ensure numerical stability while preserving the intended dynamics, we systematically tested the appropriate scaling for each force type, focusing in particular on the shape-preserving forces and the deforming overlap force.
Our method was to isolate each of these forces in simulation and determine the threshold at which the system becomes unstable. Specifically, we ran simulations with only one force active at a time and gradually increased its scaling factor until numerical instabilities emerged. We then selected the **maximum stable scaling** as the operative value for that force.
These tests were conducted using a fixed time step of 10^{-5}, with cell configurations composed of six vertices. To rigorously challenge the model, we initialized the cells in deliberately distorted and uncomfortable shapes, which are most prone to triggering instabilities. The configurations used in these tests are the same as those shown in the figures throughout this chapter, where the isolated effects of each force were illustrated following their respective introductions. This allowed us to verify that the chosen scalings are robust even under unfavorable conditions.
For the bounce overlap force, a scaling factor of 10^5 is necessary to reproduce the original bounce off dynamics used in [BC12].
Summarizing all these efforts, we arrived at the following force scalings used in the simulations:

Force type	**Scaling parameter**
Area force	$\alpha_A =$ _____
Edge force	$\alpha_E =$ _____
Interior angle force	$\alpha_I =$ _____
Deforming overlap force	$\alpha_{\hat{O}} =$ _____
Bounce overlap force	$\alpha_{\bar{O}} = 10^5$

Table 1: Scaling parameters for different force types

These scaling par ameters serve as the foundation for the complete DF SDE model, which we now introduce.

Definition 3.15. The DF SDE
We define the vectors

$$e^x_{N_V} = (1,0,1,0,\ldots,1,0)^T, \quad e^y_{N_V} = (0,1,0,1,\ldots,0,1)^T \in \mathbb{R}^{2N_V},$$

which allow us to distribute a two dimensional Brownian motion $\mathrm{d}\vec{B}^{\,i} = (\mathrm{d}B^{i,x}, \mathrm{d}B^{i,y})^T$ to the x and y coordinates of a cell's vertices, respectively. The cell hardness parameter $h \in [0,1]$ is assumed to be given. The deterministic part $F^i(\vec{C})$ of the **DF SDE** is given by

$$\mathbf{F}^i : (\mathbb{R}^{2N_V})^{N_C} \to \mathbb{R}^{2N_V}$$

$$\mathbf{F}^i(\vec{C}) = \alpha_A F_2^{(A)}(C_i) + \alpha_E F_2^{(E)}(C_i) + \alpha_I F_2^{(I)}(C_i) + (1-h)\alpha_{\hat{O}} F_{1,i}^{(\hat{O})}(\vec{C}) + h\alpha_{\bar{O}} F_i^{(\bar{O})}(\vec{C}) \tag{11}$$

for each cell $1 \leqslant i \leqslant N_C$.
Each scaling parameter $\alpha \geqslant 0$. Each F represents one of our forces that got defined in this chapter. For the shape preserving forces, we choose $k = 2$ as then the difference between current state and desired state influences the intensity of the force which is nice. But for the deforming overlap force, we choose $k = 1$ we want it to have a strong impact whenever a small overlap arises. This configuration seems to be the best to model the physics we would like to achieve. With that we can write down the cell wise formulated DF SDE:

$$\mathrm{d}C_i = \mathbf{F}^i(\vec{C})\mathrm{d}t + \mathrm{d}B^{i,x}\, e^x_{N_V} + \mathrm{d}B^{i,y}\, e^y_{N_V}.$$

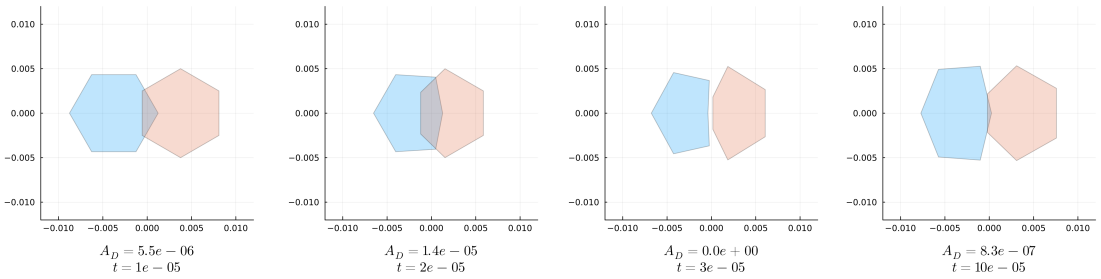

Figure 12: Click: *here*, to see the animation as a gif.

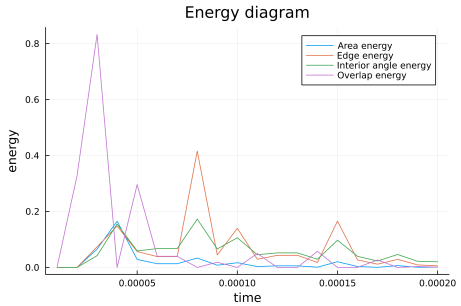

Figure 13

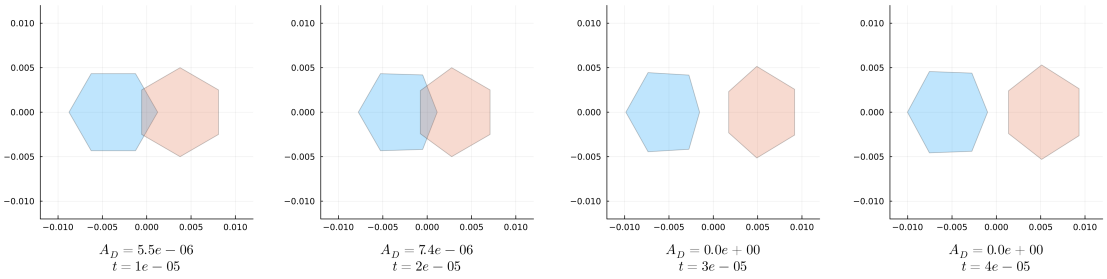

Figure 14: Click: *here*, to see the animation as a gif.

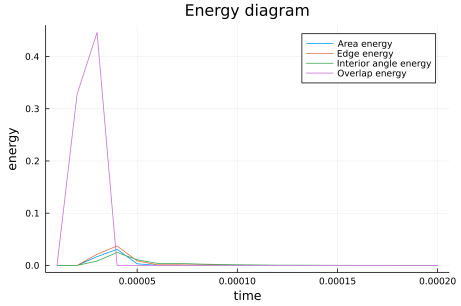

Figure 15

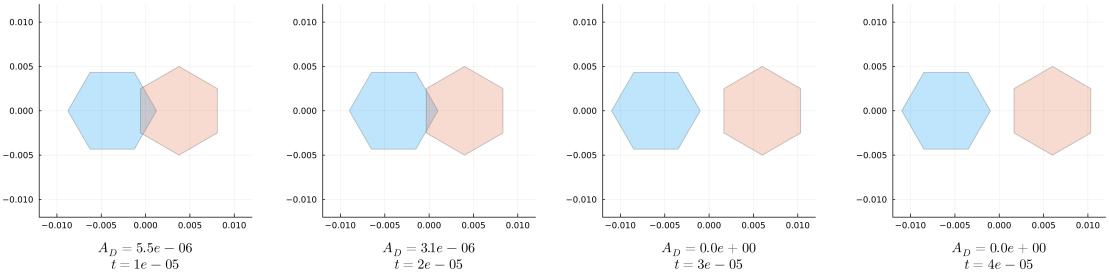

Figure 16: Click: *here*, to see the animation as a gif.

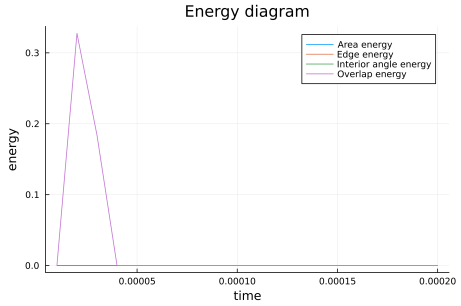

Figure 17

4 Sanity check

Having introduced our cell dynamics, we now want to take a look at the simulation results. Therefore, we aim to compare our simulation results to results from an established cell model from [BC12]. In [BC12] the diffusion dynamics of first a point particle model and second a hard sphere model is studied. Thereby, the two density distributions:

- the joint probability density function $P(\vec{X}, t)$ of the system of all cell centres $\vec{X}$ at time t,
- the marginal distribution function of the first particle $p(\vec{x}_1, t)$

play an important roll.
The joint probability density function $P(\vec{X}, t)$ is a function describing the positions of all particles in the system, while the marginal distribution function $p(\vec{x}_1, t)$ is a function describing only the position of the first particle.
It is sufficient to consider only the marginal distribution function of first particle, because all particle act similarly.
Gaining $p(\vec{x}_1, t)$ from $P(\vec{X}, t)$ is a big reduction of complexity, since we reduce from a high-dimensional PDE for P to a low-dimensional PDE for p. The marginal distribution function the of first particle can always be determined via

$$p(\vec{x}_1, t) = \int P(\vec{X}, t) \mathrm{d}\vec{x}_2 \dots \mathrm{d}\vec{x}_{N_{BC}}.$$

4.1 Reference simulations: Bruna and Chapman (2012)

The most simple model that gets considered for the diffusion dynamics of cell systems is the point particle model. Here the cells get modeled with sizeless points that perform a Brownian motion on the domain.
Since the cells do not have a real size, no interaction between the cells can occur, since they will never hit upon each other.
The paper [BC12] analyses these dynamics on the domain

$$\Omega_{BC} = [-0.5, 0.5]^2,$$

on which $N_{BC} = 400$ particles are located.
The movement of each point particle $\vec{x}_i$ in the simulation is given by the SDE

$$\mathrm{d}\vec{x}_i(t) = \sqrt{2}\mathrm{d}B_t^{(i)}, \quad 1 \leqslant i \leqslant N_{BC},$$

which describes a Brownian motion in Ω_{BC}. The reflective boundary condition on $\partial\Omega_{BC}$ is imposed. It is known, that the joint probability density of the particle system in this setup evolves according to the diffusion equation, i.e.

$$\frac{\partial P}{\partial t}(\vec{X}, t) = \Delta_{\vec{X}} P = \nabla_{\vec{X}} \cdot [\nabla_{\vec{X}} P] \tag{12}$$

inside of the domain.
Since all particles are independent, we can compute

$$P(\vec{X}) = \prod_{i=1}^{N_{BC}} p(\vec{x}_i, t). \tag{13}$$

Therefore, we obtain the marginal distribution function as

$$\frac{\partial p}{\partial t}(\vec{x}_1, t) = \Delta_{\vec{x}_1} p = \nabla_{\vec{x}_1} \cdot [\nabla_{\vec{x}_1} p]. \tag{14}$$

A next step that results in the hard sphere cell model (HSCM) is to give the cell particles a real size.
Let $0 < \epsilon \ll 1$ be the diameter of all cells that are now two dimensional discs with the same size. This changes the dynamics of the cells immense, since they now have chance to collide into each other which is a form of interaction.
The authors of [BC12] also did a simulation with the HSCM. The setting is as similar as possible to the point particle model, because a main goal of the paper was to compare the diffusion characteristics of both models. There are still $N_{BC} = 400$ cells located on the domain.

The initial condition of both models follows a two dimensional normal distribution with the addition that the distance of each cell centre to all others is at least ϵ. The used distribution $\mathcal{N}_2\left(\binom{0}{0}, \begin{pmatrix} 0.09^2 & 0 \\ 0 & 0.09^2 \end{pmatrix}\right)$ has an integral of one over Ω_{BC}.
We can compute this initial condition with Algorithm 4.1.

Algorithm 4.1. ***Computation of the initial cell system***

1. *Generate a point* $\vec{x} \sim \mathcal{N}_2\left(\binom{0}{0}, \begin{pmatrix} 0.09^2 & 0 \\ 0 & 0.09^2 \end{pmatrix}\right)$.
2. *If for all already generated centres* $\vec{x}_j : \|\vec{x} - \vec{x}_j\|_2 > \epsilon$ *is true, use* $\vec{x}$ *as the next cell centre, otherwise discard the point and restart with step 1 until* N_{BC} *cell centres are found.*

Since we do not want any overlap to occur during the whole simulation with the HSCM, the feasible domain for the whole cell system is not directly $\Omega_{BC}^{N_{BC}}$, but instead

$$\begin{aligned} \Omega_{BC}^{\epsilon} &= \Omega_1^{\epsilon} \times \ldots \times \Omega_{N_{BC}}^{\epsilon}, \\ \Omega_i^{\epsilon} &= \Omega_{BC} \backslash (\cup_{j \neq i} \mathrm{B}_{\epsilon}(\vec{x}_j)), \quad 1 \leqslant i \leqslant N_{BC}, \end{aligned}$$

where $\mathrm{B}_{\epsilon}(\vec{x}_j)$ denotes the ball around $\vec{x}_j$ with radius ϵ.
This domain prevents overlaps between the cells by not allowing each cell to drift closer than ϵ to any other cell.
HSCM cells perform the same Brownian motion as the point particles.
The next question is, how cell collisions are modelled. Unlike in our DCF model where cell interactions are modelled as forces acting inside of the domain, the cell collisions from the HSCM arise from the reflective boundary condition.
Let us assume that two cells i and j are given such that $\|\vec{x}_i - \vec{x}_j\|_2 = \epsilon$ is true. Then, both cell centres are located at the boundary $\partial\Omega_{BC}^{\epsilon}$. Here, the reflective boundary condition is still imposed and it causes both cells to bounce of from each other in the direction of the outward normal vector from the excluded area of the respectively other cell.

In [BC12] the authors managed to compute the marginal distribution function of the first particle of the HSCM. In two dimensions it is given by:

$$\frac{\partial p}{\partial t}(\vec{x}_1, t) = \nabla_{\vec{x}_1} \cdot \{\nabla_{\vec{x}_1}[p + \frac{\pi}{2}(N_{BC} - 1)\epsilon^2 p^2]\}. \tag{15}$$

We can see a connection to Equation 14. Let us define a diffusion coefficient

$$D_\epsilon(p) = 1 + \pi(N_{BC} - 1)\epsilon^2 p$$

that depends on the local partical density p and the cell diameter ϵ. For the point particles, we have $\epsilon = 0$ and $D_\epsilon(p) = 1$. Thus, we can rewrite the first marginal to

$$\frac{\partial p}{\partial t}(\vec{x}_1, t) = \nabla_{\vec{x}_1} \cdot [D_\epsilon(p)\nabla_{\vec{x}_1} p].$$

In the case of the hard spheres, where $\epsilon > 0$, we can compute

$$\begin{aligned}
\frac{\partial p}{\partial t}(\vec{x}_1, t) &= \nabla_{\vec{x}_1} \cdot [D_\epsilon(p)\nabla_{\vec{x}_1} p] \\
&= \nabla_{\vec{x}_1} \cdot [(1 + \pi(N_{BC} - 1)\epsilon^2 p)\nabla_{\vec{x}_1} p] \\
&= \nabla_{\vec{x}_1} \cdot [\nabla_{\vec{x}_1} p + \pi(N_{BC} - 1)\epsilon^2 p \nabla_{\vec{x}_1} p] \\
&= \nabla_{\vec{x}_1} \cdot [\nabla_{\vec{x}_1} p + \pi(N_{BC} - 1)\epsilon^2 \frac{1}{2}\nabla_{\vec{x}_1} p^2] \\
&= \nabla_{\vec{x}_1} \cdot [\nabla_{\vec{x}_1}(p + \frac{\pi}{2}(N_{BC} - 1)\epsilon^2 p^2)]
\end{aligned}$$

to recover Equation 15.
When considering $D_\epsilon(p) = 1 + \pi(N_{BC} - 1)\epsilon^2 p$ to be the diffusion coefficient, we can conclude that an increase in the number of cells N_{BC}, the cell diameter ϵ, or the local density p leads to an increased diffusion rate of the system. Overall, we conclude that the bounce effect of the HSCM enhances the diffusion rate of the system's density.
Another evidence of this behavior is shown in Figure 2 in [BC12].
Here, we can see two Monte Carlo simulations. A Monte Carlo simulation is a computational technique that uses random sampling to model and analyse complex systems or processes that are difficult to solve analytically. It repeatedly generates random inputs according to specified probability distributions and computes the resulting outcomes to estimate quantities like averages, variances, or distributions.
In our case, the Monte Carlo simulations are used to track the positions of cell centres over time. Each simulation begins from an initial configuration of cells, which is consistently generated using Algorithm 4.1. After initialization, the prescribed dynamics - either the point particle model or the hard sphere model - are applied, and the positions of the cell centres are recorded at a fixed time point, $t = 0.05$.
To visualise the results, we construct heatmaps representing the spatial distribution of cells at the final time. This is done by discretizing the domain into a uniform grid of sub squares. For each sub square, we count how many cells fall within it across all simulations. The resulting counts are normalised by dividing by the total

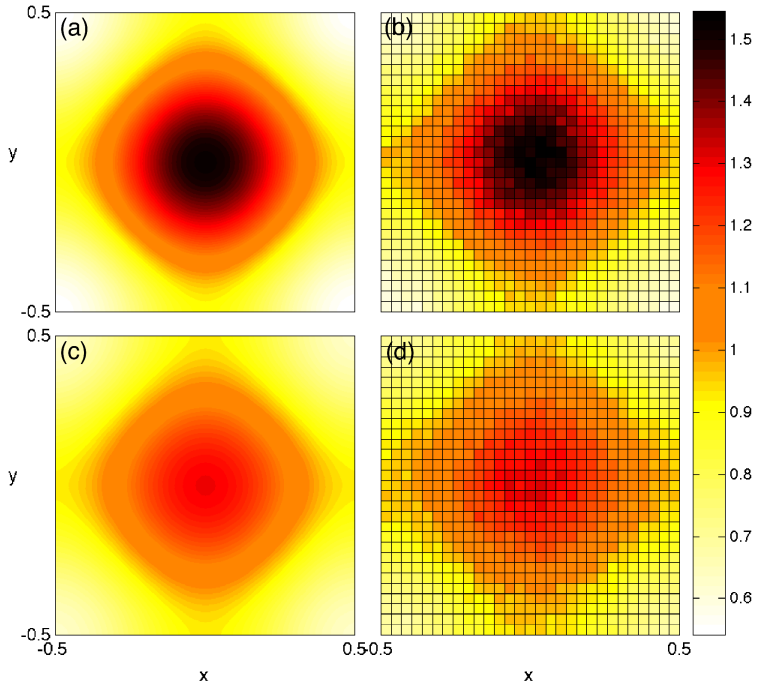

Figure 18: This figure contains the following four plots, all of them are shown at time $t = 0.05$:
(a) shows the solution of the linear diffusion equation 12 for point particles.
(b) shows the histogram of a Monte Carlo simulation of the point particle model.
(c) shows the solution of the nonlinear diffusion equation 15 for finite-sized particles.
(d) shows the histogram of a Monte Carlo simulation of the HSCM.
For the two lower plots (c) and (d), a higher diffusion rate can be observed. The Monte Carlo simulations used 10^4 simulation runs each with a time step size of 10^-5.

number of cells N_C, the number of simulations, and the area of a sub square. This normalisation ensures that the heatmap represents a probability density, satisfying the mass conservation condition:

$$\sum_{i \in \text{ sub squares}} \text{value}_i \cdot \text{area}_i = 1.$$

This approach provides a smooth estimate of the empirical cell density, allowing direct comparison with the corresponding solutions of the diffusion equations.
Figure 18 shows the discussed graphic from [BC12].

4.2 Reproduction of reference results

Before running our new dynamics that include cell flexibility, we first want to guarantee that the simulations are running in the correct setup. Therefore, we started with recreating the Monte Carlo simulation for the point particles. I always fixed the color scale to be the same as in [BC12] in order to gain comparability. The simulation parameters are the same as in [BC12].
All of our simulations run in the Julia programming language. There, we used the

package 'DifferentialEquations.jl' with its structure 'SDEProblem()' and then solved it with the package inbuilt Euler Maruyama scheme that uses a constant time step size.
I employed a callback function that was triggered after each simulation step to implement a reflective boundary condition. Whenever a particle moved outside the domain, it was relocated to the position within the domain such that its distance to the domain boundary remained unchanged, effectively reflecting the particle off the boundary.
Beside of this, all particles moved according to the two dimensional Brownian motion

$$\mathrm{d}\vec{x}_i(t) = \sqrt{2}\mathrm{d}\vec{B}^{\,i}, \quad 1 \leqslant i \leqslant N_C.$$

Figure **??** shows the evolution of the particle density in terms of heatmaps for different time steps. The results of our Monte Carlo simulation appear to be in good agreement with those of Bruna and Chapman, suggesting that our approach is robust and accurate.
Next, we consider the HSCM and run the Monte Carlo simulation for a cell diameter of $\epsilon = 0.01$. Figure **??** shows the density evolution of the HSCM.

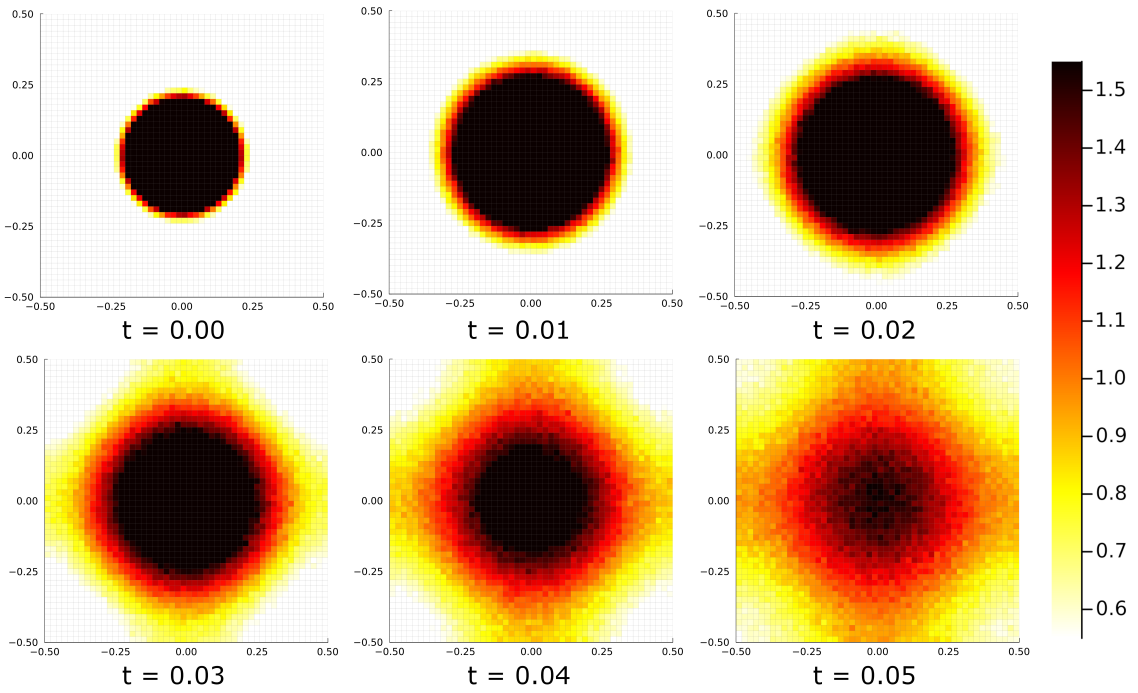

Figure 19: Heatmaps of a Monte Carlo simulation of the point particle model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. ppHeatmaps50.

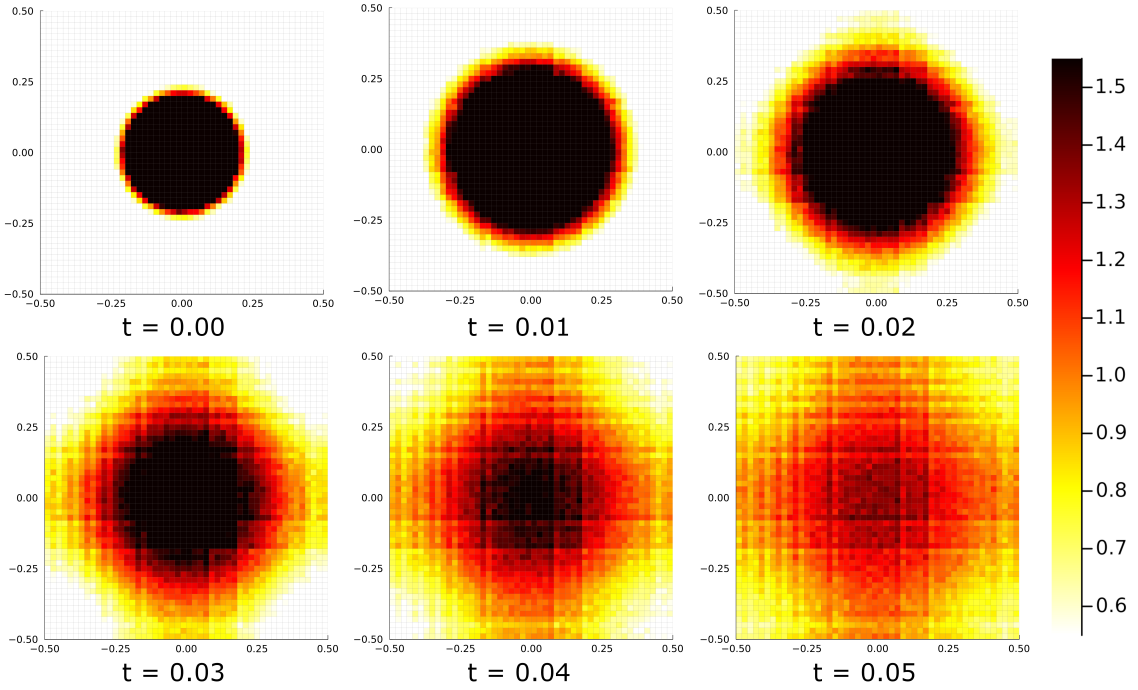

Figure 20: Heatmaps of a Monte Carlo simulation of the hard sphere cell model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. hscmHeatmapscallback50.

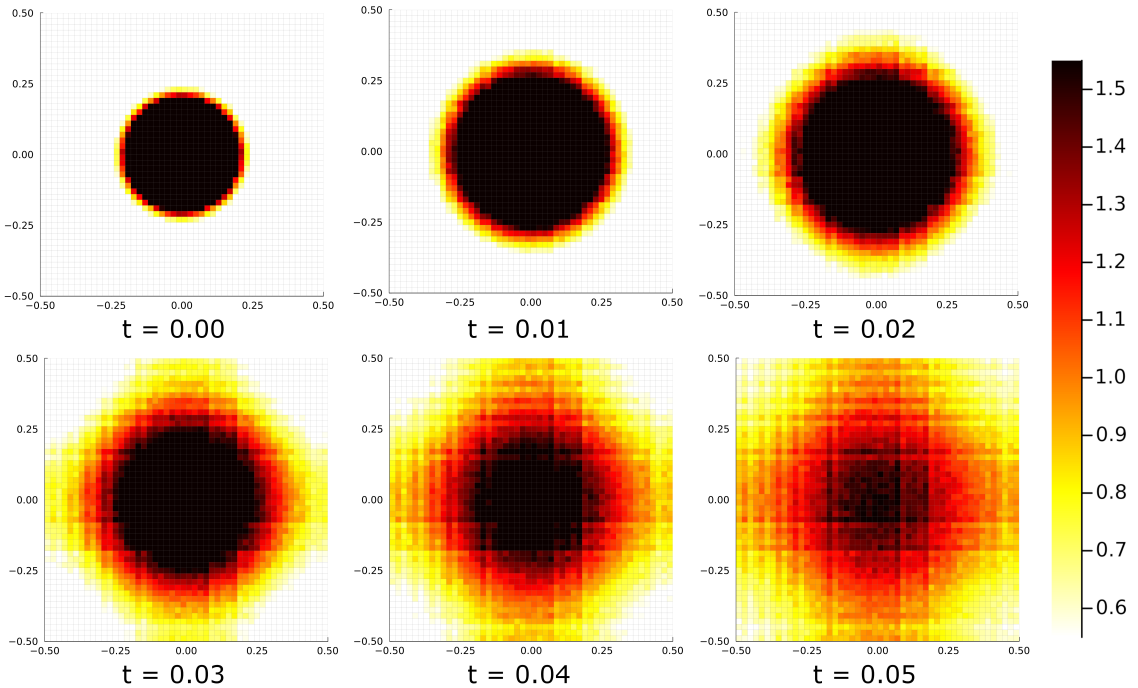

Figure 21: Heatmaps of a Monte Carlo simulation of the point particle model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. hscmHeatmapsbillard50.

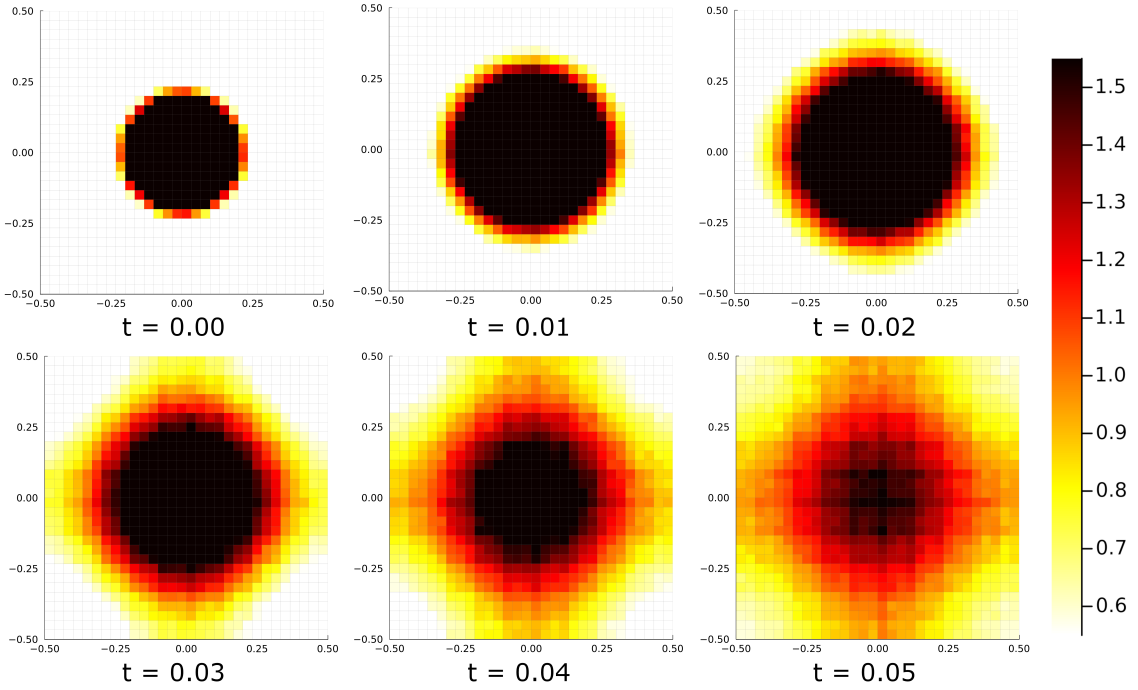

Figure 22: Heatmaps of a Monte Carlo simulation of the point particle model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. ppHeatmaps30.

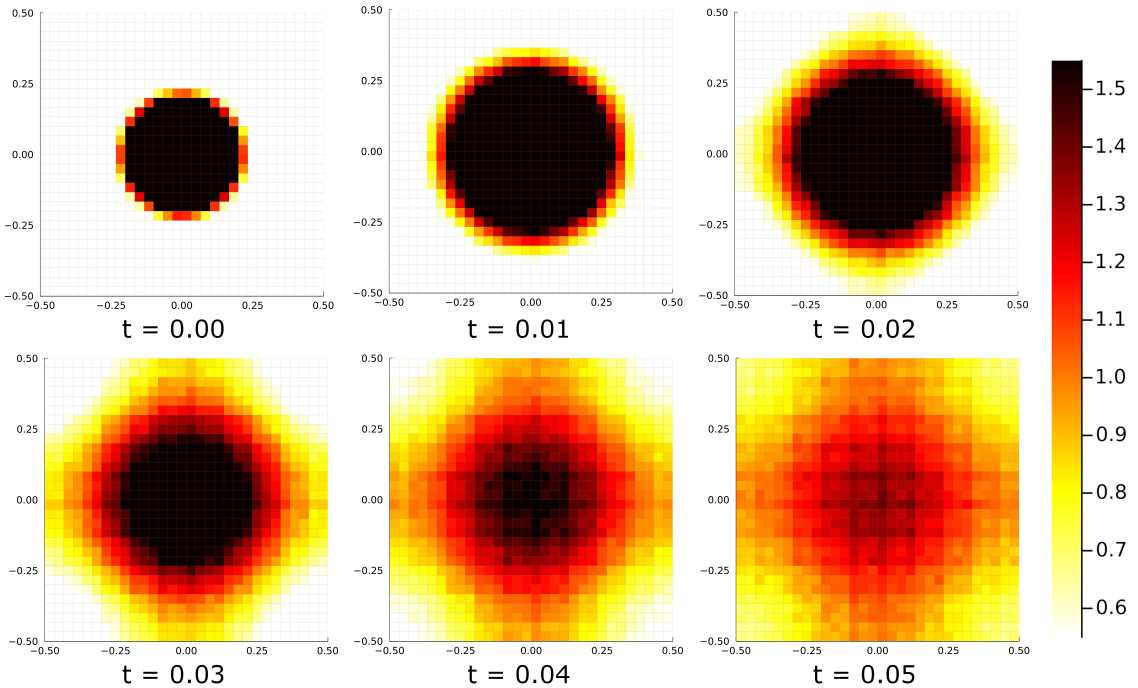

Figure 23: Heatmaps of a Monte Carlo simulation of the hard sphere cell model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. hscmHeatmapscallback30.

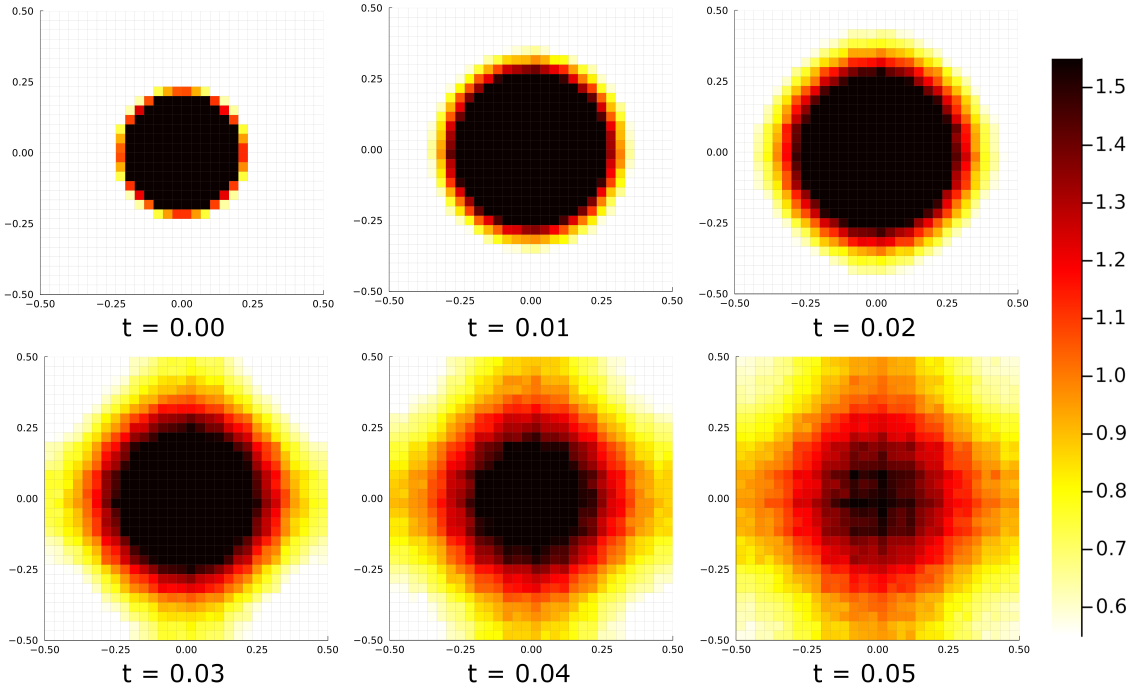

Figure 24: Heatmaps of a Monte Carlo simulation of the point particle model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. hscmHeatmapsbillard30.

5 Density computations

In the previous chapter, we studied Monte Carlo simulations of our DF model and visualised the resulting particle distributions as heatmaps at different time points. These simulations provided valuable insight into the statistical behavior of the system for finite numbers of cells.
In this chapter, we pursue a different approach: instead of extracting approximate density fields from stochastic simulations, we aim to compute the underlying density distribution function ρ that describes the limiting behavior of the system as the number of cells $N_C \to \infty$. More precisely, we seek the density ρ such that the limiting measure μ satisfies $\mathrm{d}\mu = \rho(x)\mathrm{d}x$. Of course, this density will depent on the desired cell state, i.e. the number of vertices and its shape, as well as on the forces we apply on the vertices as the dynamic and their scalings.
As a starting point for this analytical treatment, we consider the empirical measure μ^{N_C}, which encodes the particle configuration at finite N_C, and study its convergence to the continuous measure μ.
The empirical measure $\mu^{N_C} \in \mathcal{P}(\mathbb{R}^2)$ is the starting point of this computation.
Let $\{\vec{x}_i\}_{i=1}^{N_C} \subset \mathbb{R}^2$ be the set of all cells' centre points. We define μ^{N_C} as:

$$\mu^{N_C} : \mathcal{B}(\mathbb{R}^2) \to [0,1]$$
$$\mu^{N_C}(A) = \frac{1}{N_C}\sum_{i=1}^{N_C} \delta_{\vec{x}_i(t)}(A),$$

where $\mathcal{B}(\mathbb{R}^2)$ is the Borel sigma-algebra on $\mathbb{R}^2$ and $\delta_{\vec{x}_i(t)}$ denotes the Dirac measure:

$$\delta_{\vec{x}_i(t)} : \mathcal{B}(\mathbb{R}^2) \to \{0,1\}$$
$$\delta_{\vec{x}_i(t)}(A) = \begin{cases} 1 & \text{if } \vec{x}_i(t) \in A, \\ 0 & \text{if } \vec{x}_i(t) \notin A. \end{cases}$$

For any test function $\phi \in C_c^\infty(\mathbb{R}^2)$, the Dirac measure satisfies

$$\int_{\mathbb{R}^2} \phi(x)\mathrm{d}\delta_{\vec{x}_i(t)}(x) = \phi(\vec{x}_i(t)).$$

For a set $A \in \mathcal{B}(\mathbb{R}^2)$, $\mu^{N_C}(A)$ is the relative proportion of the N_C particles that are located in A.
For a finite $N_C \in \mathbb{N}$, μ^{N_C} is a discrete measure that only has its mass divided on the exact particle locations. As we increase the number of particles, μ^{N_C} will spread out - having more particle locations to cover with each particle having a lower influence on the result of μ^{N_C} as we divide through N_C. This process is quite simular to the transition from a sum to an according integral:

$$\sum_{i=1}^{N_C} \frac{1}{N_C} f(x_i) \xrightarrow{N\to\infty} \int f(x)\mathrm{d}x,$$

where we can also see a transition from a discrete starting problem, having discrete points x_i, to a continious integral where $x \in (a,b)$. As μ^N is a measure that lives on

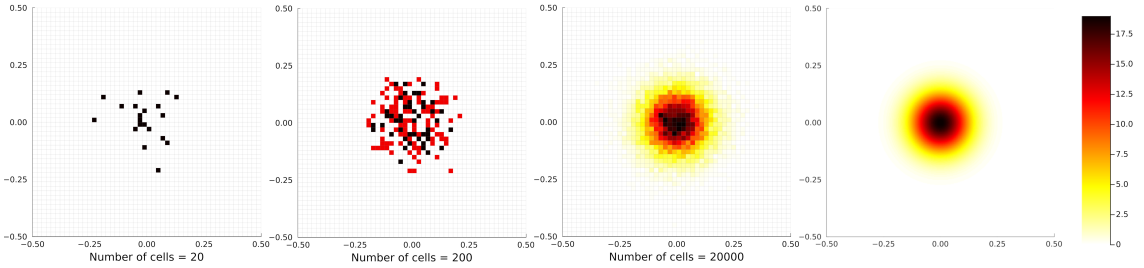

Figure 25: Visualisation of empirical measures μ^{N_C} for increasing numbers of cells, alongside the corresponding theoretical density. This visualisation illustrates the transition $\mu^{N_C} \xrightarrow{N_C \to \infty} \mu$.
All cell configurations are sampled from the same initial condition described in the previous chapter: a two-dimensional normal distribution $\mathcal{N}_2((0,0), \sigma^2 I_2)$ with $\sigma = 0.09$. In the first three subplots, the domain $[-0.5, 0.5]^2$ is discretised into square bins of side length $\frac{1}{50}$. Each bin A corresponds to a measurable set in the definition of the empirical measure $\mu^{N_C}(A) = \frac{1}{N_C} \sum_{i=1}^{N_C} \delta_{x_i}(A)$, where the color intensity encodes the number of cells in that bin.
The first subplot shows a realisation with $N_C = 20$ cells, the second with $N_C = 200$, and the third with $N_C = 20.000$. We observe that as N_C increases, the empirical measure μ^{N_C} becomes a smoother approximation of the underlying density. The color scale is fixed across all subplots to allow direct visual comparison. The fourth panel displays the exact density function of the initial distribution, $\rho(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$, with $\sigma = 0.09$.

sets $A \in \mathcal{B}(\mathbb{R}^d)$, we cannot directly plot it as a function. Instead, we try to visualise it meaningfully by using histograms as approximations in Figure 25. This is also a good connection from the previus section, where we also used histograms to show the results of the monte carlo simulations.
The distribution from the fourth subplot in Figure 25 is aimed to be computed for the dynamics of our DF cell model.
In the end, we want to achieve:

$$\mu^{N_C} \xrightarrow{N_C \to \infty} \mu$$

by letting the number of cells go to infinity.

5.1 Transition $\mu^{N_C} \xrightarrow{N_C \to \infty} \mu$

5.2 General energy computation

Lets define our cell centres with:

$$\vec{X} = (\vec{x}_1, \ldots, \vec{x}_N) \in \mathbb{R}^{2N} (\text{vector of all cell centres}),$$
$$\text{for } \vec{x}_i \in \mathbb{R}^2,\ 1 \leqslant i \leqslant N.$$

The energy that gets used for our cell dynamic shall be:

$$E : \mathbb{R}^2 \to \mathbb{R}$$
$$E(\vec{x}_i) = \frac{1}{2}|\vec{x}_i|^2.$$
$$\nabla_{(\vec{x}_i)} E(\vec{x}_i) : \mathbb{R}^2 \to \mathbb{R}^2$$
$$\nabla_{(\vec{x}_i)} E(\vec{x}_i) = |\vec{x}_i|.$$

We define the dynamic of a particle $\vec{x}_i$ via:

$$\frac{\mathrm{d}\vec{x}_i}{\mathrm{d}t} = -\nabla_{\vec{x}_i} E(\vec{x}_i) \in \mathbb{R}^2.$$

We define the probability measure:
(question: is μ defined on a single particle $[\mu^{N_C} \in \mathcal{P}(\mathbb{R}^2)]$ or on the whole particle system $[\mu^{N_C} \in \mathcal{P}(\mathbb{R}^{2N})]$)
(question: what does μ say?
Its 1 when the particle is at a given location? vs Its 1 when the particle system is at a given configuration?)

$$\mu : \mathbb{R}^2 \to [0, \infty)$$

μ is the density of cell system. μ^N is the empirical measure. It takes a subset $A \subset \mathbb{R}^2$ as an argument and gives the relative number of particles that are inside of A.
Let $\phi \in C_c^\infty(\mathbb{R}^2, \mathbb{R})$ (??) be a test function. Its gradient field is $\nabla\phi : \mathbb{R}^2 \to \mathbb{R}^2$. We compute:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\int \phi \mathrm{d}\mu^N &= \frac{\mathrm{d}}{\mathrm{d}t}(\frac{1}{N_C}\sum_{i=1}^{N_C}\phi(\vec{x}_i)) \\
&= -\frac{1}{N_C}\sum_{i=1}^{N_C}\nabla\phi(\vec{x}_i)\cdot\nabla E(\vec{x}_i) \\
&= -\frac{1}{N_C}\sum_{i=1}^{N_C}\int \nabla\phi(x)\cdot\nabla E(x)\mathrm{d}\delta_{\vec{x}_i} \\
&= -\int \nabla\phi(x)\cdot\nabla E(x)\mathrm{d}\mu^N\mathrm{d}x \\
&= \int \phi(x)\nabla\cdot(\mu^N(\nabla E(x)))\mathrm{d}x
\end{aligned}$$

$$\Rightarrow 0 = \partial_t\rho - \nabla(\rho_v),$$

where ρ is the density function of μ such that

$$\mu(dx) = \rho(x)dx.$$

question: what is the space we integrate above? (i gues $\mathbb{R}^{2N}$)

6 Outlook

An interesting extension of the current model would involve assigning individual desired states to each cell, in contrast to the uniform desired state used throughout this study. This modification would naturally lead to cell-specific energies and corresponding forces, as both would depend on the unique desired configuration of each cell. Incorporating such heterogeneity could allow the model to capture more complex biological behaviors, such as differentiation, cell-type-specific migration, or adaptive responses to environmental cues.

Statement of authorship

I hereby declare that I have written this thesis (*Derivation and study of a non-confluent model for deformable cells*) under the supervision of Jun.-Prof. Dr. Markus Schmidtchen independently and have listed all used sources and aids. I am submitting this thesis for the first time as part of an examination. I understand that attempted deceit will result in the failing grade „not sufficient“ (5.0).

Tim Vogel
Dresden, September 1, 2025
Technische Universität Dresden
Matriculation Number: 4930487

References

[BC12] Maria Bruna and S. Jonathan Chapman. Excluded-volume effects in the diffusion of hard spheres. *Phys. Rev. E*, 85:011103, Jan 2012.

[Sho14] ShoelaceFormula. Green's theorem and area of polygons. blogoverflow, June 2014. Published by: apnorton. URL: https://math.blogoverflow.com/2014/06/04/greens-theorem-and-area-of-polygons/. LAst accessed on 23.11.2023.

[Sho22] ShoelaceIllustration. Deriving the trapezoid formula. URL: https://commons.wikimedia.org/wiki/File:Trapez-formel-prinz.svg, January 2022. Published by user 'Ag2gaeh'. Last accessed on 23.11.2023.

[Vog23] Tim Vogel. Modelling of cells and their dynamics. 2023.