Technische Universität Dresden • Faculty of Mathematics

Diffusitivity of 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: July 1, 2025

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

Contents

1 TODOs

- add left(, right) where it is necessary
- be consistent in the notation $\vec{v} = (v^1, v^2)$, dont use (v_x, v_y) or something like that. Look at definition of $arctan(x, y) -> arctan(v^1, v^2)$
- 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.

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 counterclockwise.
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

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

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 $\vec{B}^{\,i}$.

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.

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 $C = (\vec{v}_1, \dots, \vec{v}_N)$ be a DF cell with $\vec{v}_j = (v_j^x, v_j^y)^T$ for $j = 1, \dots, N$. 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.

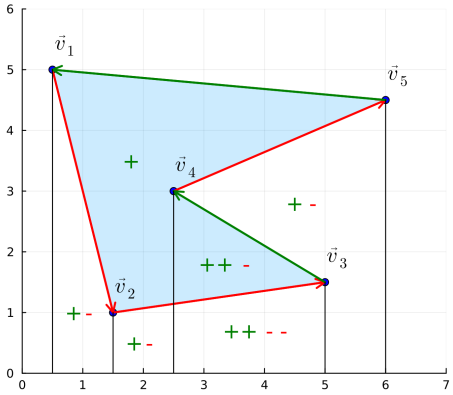

Figure 1: This figure shows a geometrical interpretation of the shoelace formula. The green arrows, which point from right to left, represent positive trapezoidal areas that contribute positively to the total area of the polygon. In contrast, the red arrows point from left to right and represent negative 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]

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* $\overrightarrow{\vec{v}_j\, \vec{v}_{j+1}}$ *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*

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

□

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 $A : (\mathbb{R}^2)^{N_V} \to \mathbb{R}_{\geqslant 0}$, used to keep the cells at a constant volume, reads

$$A(C_i) = \frac{1}{2}|A_d - A_{C_i}|^2, \tag{1}$$

where A_d is the desired cell area of all cells and A_{C_i} is the current area of cell i.

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 gradient $\nabla_{\vec{v}_j} A(C)$ with respect to the jth vertex of cell C is given by

$$\nabla_{\vec{v}_j} A(C) = \frac{1}{2}(A_C - A_d) \begin{pmatrix} v^y_{j+1} - v^y_{j-1} \\ v^x_{j-1} - v^x_{j+1} \end{pmatrix},$$

where $\vec{v}_j = (v^x_j, v^y_j)^T$.

Thus, the area force $F^{(A)}_j : (\mathbb{R}^2)^{N_V} \to \mathbb{R}^2$ that gets applied on $\vec{v}_j$ is given by

$$F^{(A)}_j(C) = -\nabla_{\vec{v}_j} A(C) = \frac{1}{2}(A_d - A_C) \begin{pmatrix} v^y_{j+1} - v^y_{j-1} \\ v^x_{j-1} - v^x_{j+1} \end{pmatrix}. \tag{2}$$

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

$$\begin{aligned}
\nabla_{\vec{v}_j} A(C) &= \nabla_{\vec{v}_j} \frac{1}{2}|A_d - A_C|^2 \\
&= (A_d - A_C)\nabla_{\vec{v}_j}(A_d - A_C) \\
&= (A_d - A_C)\nabla_{\vec{v}_j}(-A_C) \\
&= (A_d - A_C)\nabla_{\vec{v}_j}(-\frac{1}{2}\sum_{k=1}^{N}(v^x_k v^y_{k+1} - v^x_{k+1} v^y_k)) \\
&= -\frac{1}{2}(A_d - A_C) \begin{pmatrix} \partial_{v^x_j}(v^x_j v^y_{j+1} - v^x_j v^y_{j-1}) \\ \partial_{v^y_j}(v^x_{j-1} v^y_j - v^x_{j+1} v^y_j) \end{pmatrix} \\
&= \frac{1}{2}(A_C - A_d) \begin{pmatrix} v^y_{j+1} - v^y_{j-1} \\ v^x_{j-1} - v^x_{j+1} \end{pmatrix}
\end{aligned}$$

Remember that A_d is just an independent constant. □

It is also valid to write $F^{(A)}_j(\vec{C})$ instead of $F^{(A)}_j(C)$, since C is included in $\vec{C}$.
Figures 2 and 3 illustrate how the area force acts on a cell to either expand or contract it toward the desired target area.

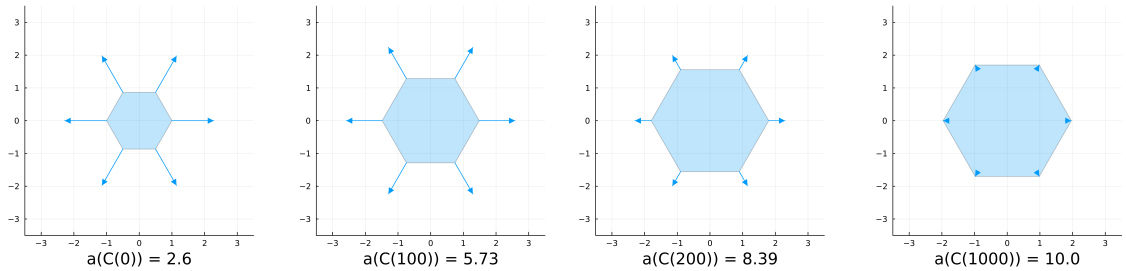

Figure 2: The figure displays the solution of Model Initially, the cell has an area of approximately 2.6. Arrows originating from each vertex represent the forces acting on the vertices at the corresponding time. The area force acts by pushing the vertices outward from the cell center, resulting in an increase in area. The computed areas at each time step are indicated below the respective diagrams. We can observe that the forces decrease as the actual cell area gets closer to its desired state. Once the target area of $A_d = 10.0$ is reached, the force vanishes and the system reaches a steady state.

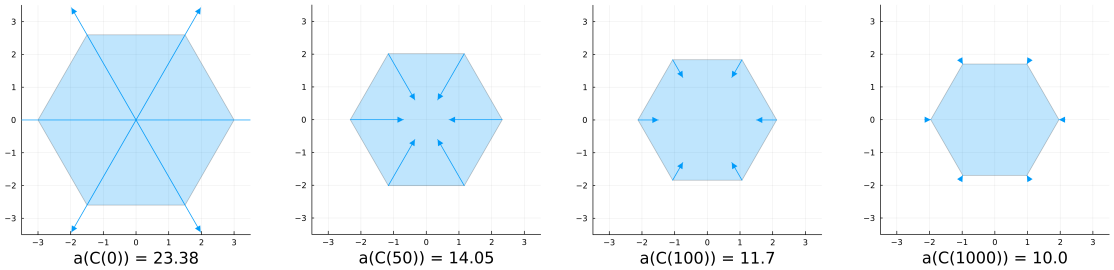

Figure 3: Similar to Figure 2, this image shows a cell that develops according to the area force. In contrast to the previous illustration, the initial area is now larger than the desired target area. As a result, the area force needs to reverse its direction to shrink the cell toward the target area $A_d = 10.0$ This outcome is consistently demonstrated across the four diagrams.

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 $E : (\mathbb{R}^2)^{N_V} \to \mathbb{R}_{\geqslant 0}$, used to keep the edges at a constant length, reads

$$E(C) = \sum_{j=1}^{N_V} \frac{1}{2} |E_d^j - E_C^j|^2, \tag{3}$$

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

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 $F_j^{(E)} : (\mathbb{R}^2)^{N_V} \to \mathbb{R}^2$ acting on $\vec{v}_j$ of the given cell is described by the formula

$$\begin{aligned} F_j^{(E)}(C) &= -\nabla_{\vec{v}_j} E(C) \\ &= \frac{E_d^{j-1} - E_C^{j-1}}{E_C^{j-1}} \begin{pmatrix} v_j^x - v_{j-1}^x \\ v_j^y - v_{j-1}^y \end{pmatrix} + \frac{E_d^j - E_C^j}{E_C^j} \begin{pmatrix} v_j^x - v_{j+1}^x \\ v_j^y - v_{j+1}^y \end{pmatrix} \end{aligned} \tag{4}$$

Proof.

$$\begin{aligned} \nabla_{\vec{v}_j} E(C) &= \nabla_{\vec{v}_j} \sum_{j=1}^{N_V} \frac{1}{2} |E_d^j - E_{C_i}^j|^2 \\ &= \nabla_{\vec{v}_j} \frac{1}{2} |E_d^{j-1} - E_{C_i}^{j-1}|^2 + \nabla_{\vec{v}_j} \frac{1}{2} |E_d^j - E_{C_i}^j|^2 \end{aligned}$$

For the first summand, we can compute

$$\begin{aligned}
\nabla_{\vec{v}_j}\frac{1}{2}|E_d^{j-1}-E_{C_i}^{j-1}|^2 &= (E_d^{j-1}-E_C^{j-1})\nabla_{\vec{v}_j}(E_d^{j-1}-E_C^{j-1}) \\
&= (E_d^{j-1}-E_C^{j-1})(-\nabla_{\vec{v}_j}\|\vec{v}_{j-1}-\vec{v}_j\|_2) \\
&= (E_d^{j-1}-E_C^{j-1})(-\nabla_{\vec{v}_j}[(v_{j-1}^x-v_j^x)^2+(v_{j-1}^y-v_j^y)^2]^{\frac{1}{2}}) \\
&= (E_d^{j-1}-E_C^{j-1})\nabla_{\vec{v}_j}(-E_C^{j-1}) \\
&= (E_d^{j-1}-E_C^{j-1})(-\frac{1}{2\|\vec{v}_{j-1}-\vec{v}_j\|_2}\nabla_{\vec{v}_j}[(v_{j-1}^x-v_j^x)^2+(v_{j-1}^y-v_j^y)^2]) \\
&= (E_d^{j-1}-E_C^{j-1})\left(-\frac{1}{2E_C^{j-1}}\begin{pmatrix}\partial_{v_j^x}(v_{j-1}^x-v_j^x)^2\\ \partial_{v_j^y}(v_{j-1}^y-v_j^y)^2\end{pmatrix}\right) \\
&= (E_d^{j-1}-E_C^{j-1})\left(-\frac{1}{2E_C^{j-1}}\begin{pmatrix}-2(v_{j-1}^x-v_j^x)\\ -2(v_{j-1}^y-v_j^y)\end{pmatrix}\right) \\
&= \frac{E_C^{j-1}-E_d^{j-1}}{E_C^{j-1}}\begin{pmatrix}v_j^x-v_{j-1}^x\\ v_j^y-v_{j-1}^y\end{pmatrix}
\end{aligned}$$

For the second summand we get:

$$\begin{aligned}
\nabla_{\vec{v}_j}\frac{1}{2}|E_d^{j}-E_{C_i}^{j}|^2 &= (E_d^{j}-E_C^{j})\nabla_{\vec{v}_j}(E_d^{j}-E_C^{j}) \\
&= (E_d^{j}-E_C^{j})(-\frac{1}{2\|\vec{v}_{j}-\vec{v}_{j+1}\|_2}\nabla_{\vec{v}_j}[(v_{j}^x-v_{j+1}^x)^2+(v_{j}^y-v_{j+1}^y)^2]) \\
&= (E_d^{j}-E_C^{j})\left(-\frac{1}{2E_C^{j}}\begin{pmatrix}\partial_{v_j^x}(v_{j}^x-v_{j+1}^x)^2\\ \partial_{v_j^y}(v_{j}^y-v_{j+1}^y)^2\end{pmatrix}\right) \\
&= (E_d^{j}-E_C^{j})\left(-\frac{1}{2E_C^{j}}\begin{pmatrix}2(v_{j}^x-v_{j+1}^x)\\ 2(v_{j}^y-v_{j+1}^y)\end{pmatrix}\right) \\
&= \frac{E_C^{j}-E_d^{j}}{E_C^{j}}\begin{pmatrix}v_j^x-v_{j+1}^x\\ v_j^y-v_{j+1}^y\end{pmatrix}
\end{aligned}$$

□

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.

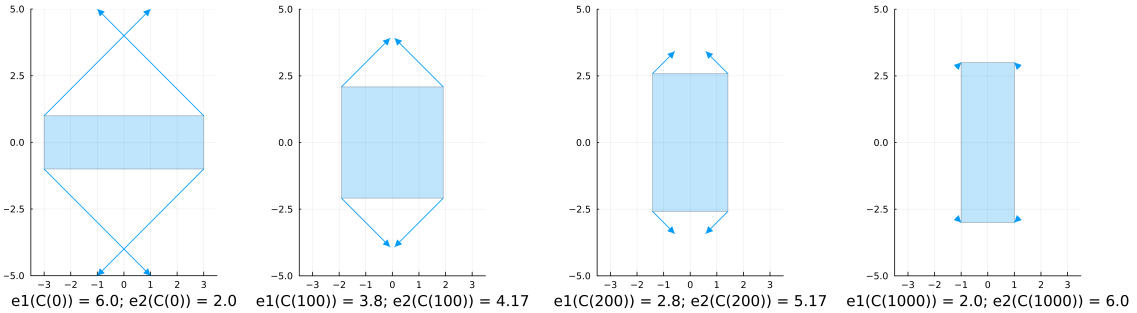

Figure 4: The diagram illustrates the edge force acting on a DF cell. At time $t = 0$, the cell is shaped as a rectangle $[-3, 3] \times [-1, 1]$, giving horizontal edges a length of 6 and vertical edges a length of 2. The corresponding edge lengths at the different time steps are annotated below each diagram. The target configuration is the rectangle $[-1, 1] \times [-3, 3]$, implying that the horizontal edges need to contract while the vertical edges must stretch. This transformation is clearly observable in the progression of the diagrams.

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 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

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

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

$$I(C) = \sum_{j=1}^{N_V} \frac{1}{2} |I_d^j - I_C^j|^2, \tag{5}$$

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

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

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 $F_j^{(I)}(C) : (\mathbb{R}^2)^{N_V} \to \mathbb{R}^2$ that gets applied on vertex $\vec{v}_j$ of cell C is given by

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

Proof.
We are looking for

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

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

$$\begin{aligned}\nabla_{\vec{v}_j} I_j(C) &= \nabla_{\vec{v}_j} \sum_{j=1}^{N_V} \frac{1}{2}|I_d^j - I_{C_i}^j|^2 \\ &= \nabla_{\vec{v}_j} \frac{1}{2}|I_d^{j-1} - I_{C_i}^{j-1}|^2 + \nabla_{\vec{v}_j} \frac{1}{2}|I_d^j - I_{C_i}^j|^2 + \nabla_{\vec{v}_j} \frac{1}{2}|I_d^{j+1} - I_{C_i}^{j+1}|^2\end{aligned}$$

First, we will focus on the computation of $\nabla_{\vec{v}_j} \frac{1}{2}|I_d^j - I_{C_i}^j|^2$.

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

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* $\arctan2(\vec{v}_{j-1} - \vec{v}_j)$ *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.

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

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

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

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:

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

For the neighboring vertices, we compute

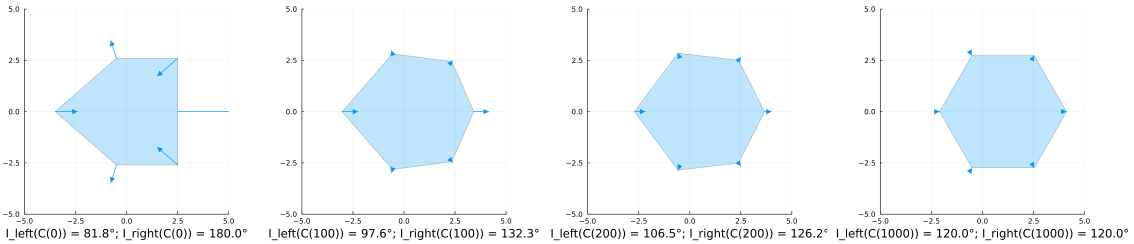

Figure 5: The figure illustrates the action of the interior angle force on the vertices of a DF cell. The initial state, shown in the first plot, must be mirrored horizontally to achieve the desired state depicted in the final plot. This reshaping shows on the one hand that the dynamic is capable of decreasing interior angles from 270° to 90° and on the other hand it can also increase interior angles from 90° to 270°. Below each chart, we can see the current interior angles of the two vertices that have the y value zero.

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

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

□

Figure 5 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.
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_k be two cells from the system $\vec{C}$ and $\Omega_{i,k}$ be the set of all overlaps that appear between C_i and C_k, like explained above. Then, the total overlap energy of the cell system is given by the formula

$$O(\vec{C}) = \sum_{i=1}^{N_C} \left(\sum_{k=i+1}^{N_C} \left(\sum_{D_l \in \Omega_{i,k}} \frac{1}{2} |A_{D_l}|^2 \right) \right), \tag{7}$$

where A_{D_l} is the area of the overlap D_l.

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.

Proposition 3.11. ***Overlap force***
The overlap force that acts on the vertex $\vec{v}_j$ of cell i is given by

$$F_{i,j}^{(O)}(\vec{C}) = \sum_{k=1, k\neq i}^{N_C} \left(\sum_{D_l \in \Omega_{i,k}} -\frac{1}{2} A_{D_l} \begin{pmatrix} d_{j+1}^{D_l,2} - d_{j-1}^{D_l,2} \\ d_{j-1}^{D_l,1} - d_{j+1}^{D_l,1} \end{pmatrix} \right), \tag{8}$$

where $\Omega_{i,k}$ is the set of all overlaps that arise between the cells i and k, $\vec{d}_{j-1}^{D_l} = (d_{j-1}^{D_l,1}, d_{j-1}^{D_l,2})^T$ and $\vec{d}_{j+1}^{D_l} = (d_{j+1}^{D_l,1}, d_{j+1}^{D_l,2})^T$ are the neighboring vertices of $\vec{v}_j$ in the overlap D_l and A_{D_l} is the area of the overlap D_l.

Proof.
The formula arises from the sums from the overlap energy and the gradient from the area force in Proposition 3.3, where the desired area is set to zero.

Figure 6 illustrates the interaction between two overlapping cells, highlighting the effect of the overlap force on their vertices.

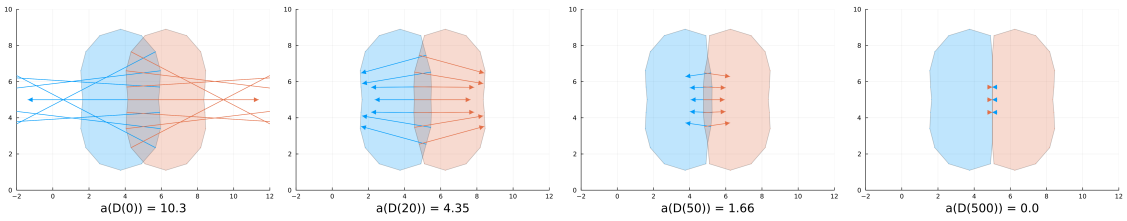

Figure 6: This figure demonstrates how the overlap force acts on two overlapping DF cells. The blue arrows indicate the forces acting on the blue cell, while the red arrows represent those acting on the red cell. The current overlap areas at each time step are displayed below the corresponding diagrams. On each consecutive diagram, we can see that the overlap gets reduced, until the area of the overlap is zero.

3.5 A simulation run

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.

$$\frac{\partial P}{\partial t}(\vec{X}, t) = \Delta_{\vec{X}} P = \nabla_{\vec{X}} \cdot [\nabla_{\vec{X}} P] \tag{9}$$

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

$$P(\vec{X}) = \prod_{i=1}^{N_{BC}} p(\vec{x}_i, t). \tag{10}$$

Therefore, we obtain the marginal distribution function as

$$\frac{\partial p}{\partial t}(\vec{x}_1, t) = \Delta_{\vec{x}_1} p = \nabla_{\vec{x}_1} \cdot [\nabla_{\vec{x}_1} p]. \tag{11}$$

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:

$$\frac{\partial p}{\partial t}(\vec{x}_1, t) = \nabla_{\vec{x}_1} \cdot \{\nabla_{\vec{x}_1}[p + \frac{\pi}{2}(N_{BC} - 1)\epsilon^2 p^2]\}. \tag{12}$$

We can see a connection to Equation 11. 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 12.
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

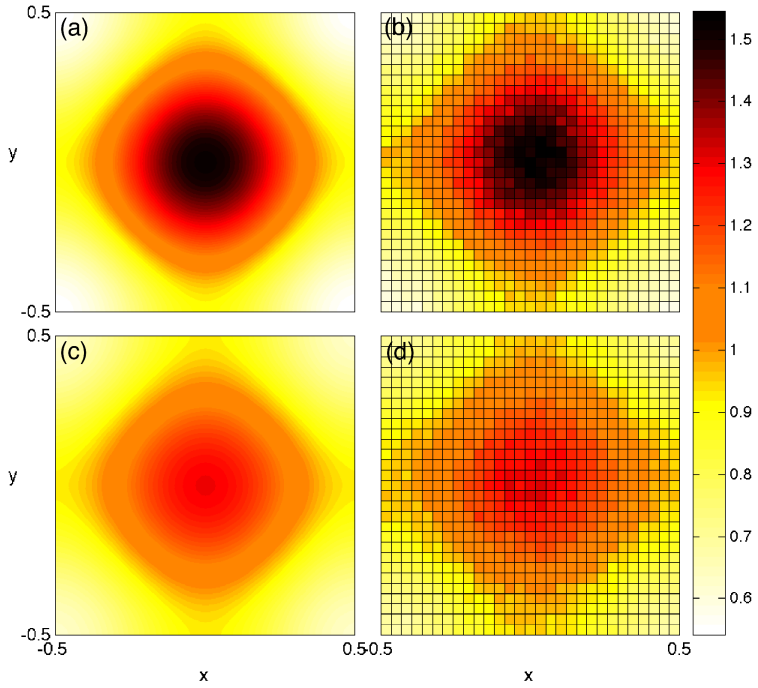

Figure 7: This figure contains the following four plots, all of them are shown at time $t = 0.05$:
(a) shows the solution of the linear diffusion equation 9 for point particles.
(b) shows the histogram of a Monte Carlo simulation of the point particle model.
(c) shows the solution of the nonlinear diffusion equation 12 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 7 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.

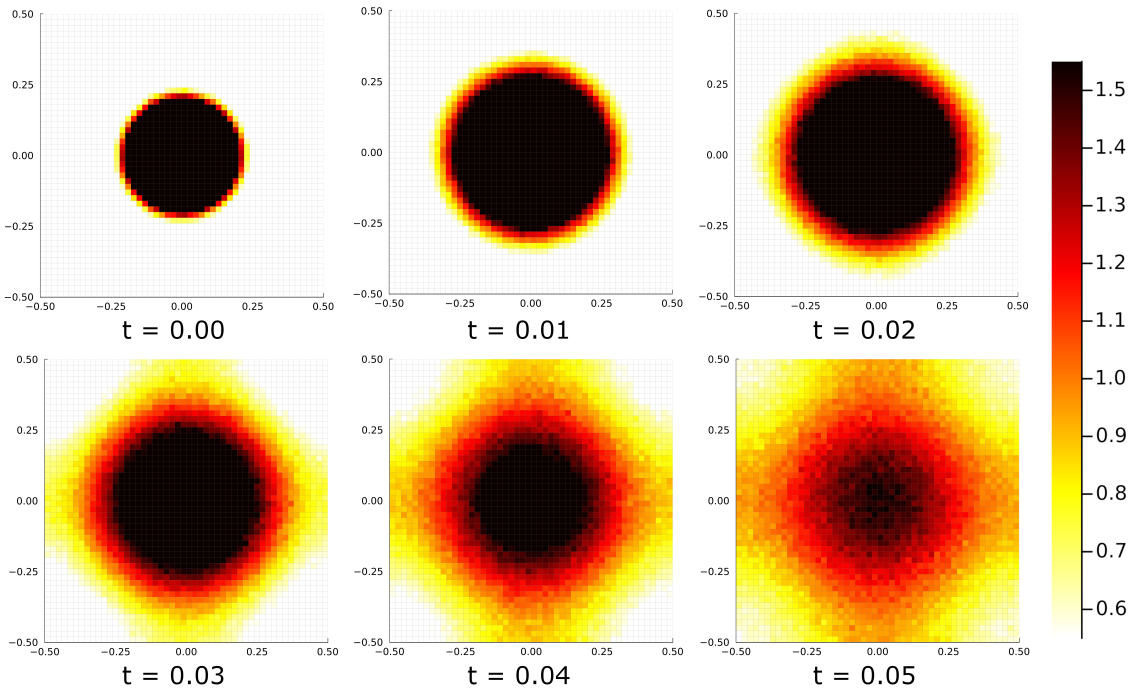

Figure 8: Heatmaps of a Monte Carlo simulation of the point particle model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. ppHeatmaps50.

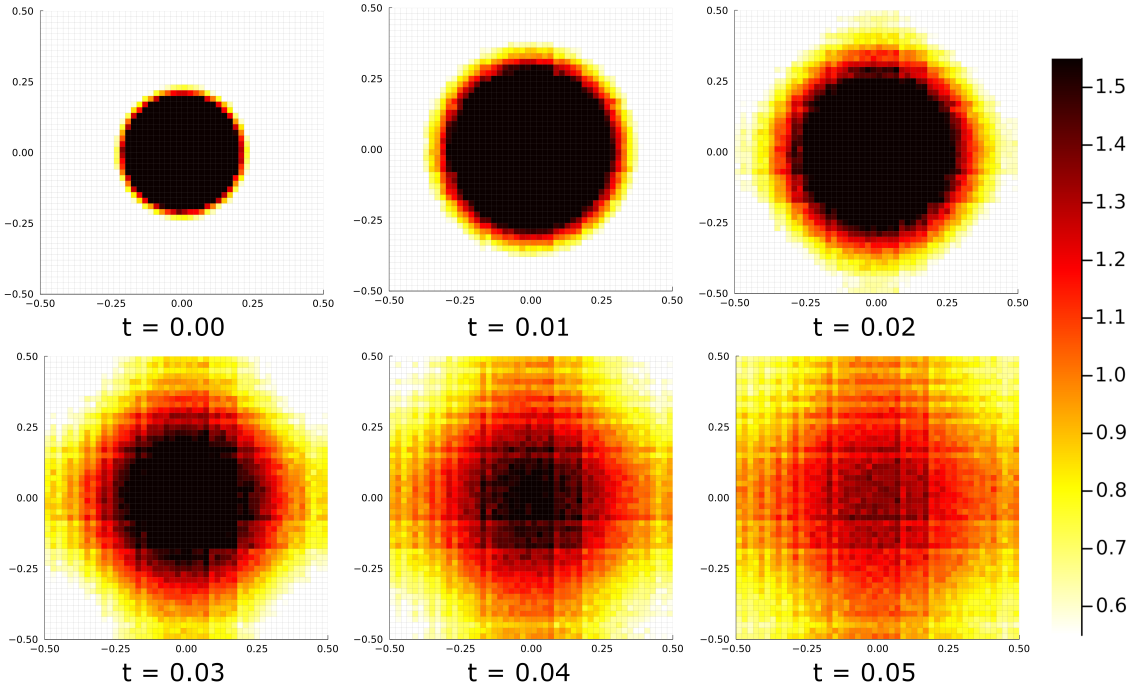

Figure 9: Heatmaps of a Monte Carlo simulation of the hard sphere cell model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. hscmHeatmapscallback50.

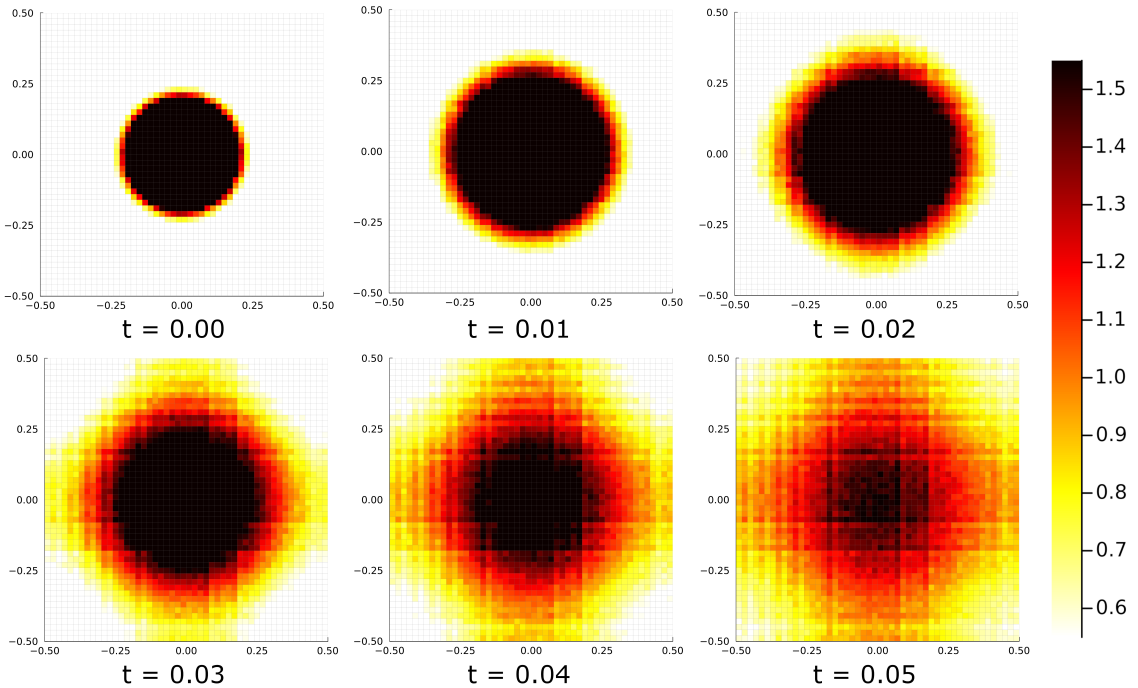

Figure 10: Heatmaps of a Monte Carlo simulation of the point particle model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. hscmHeatmapsbillard50.

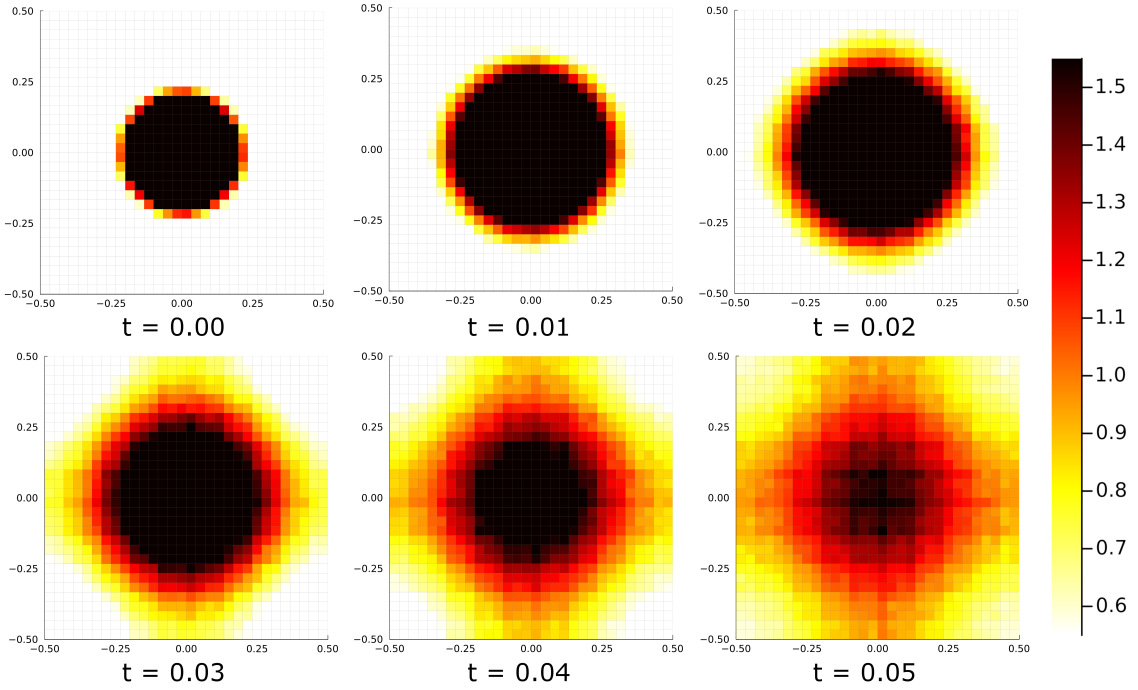

Figure 11: Heatmaps of a Monte Carlo simulation of the point particle model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. ppHeatmaps30.

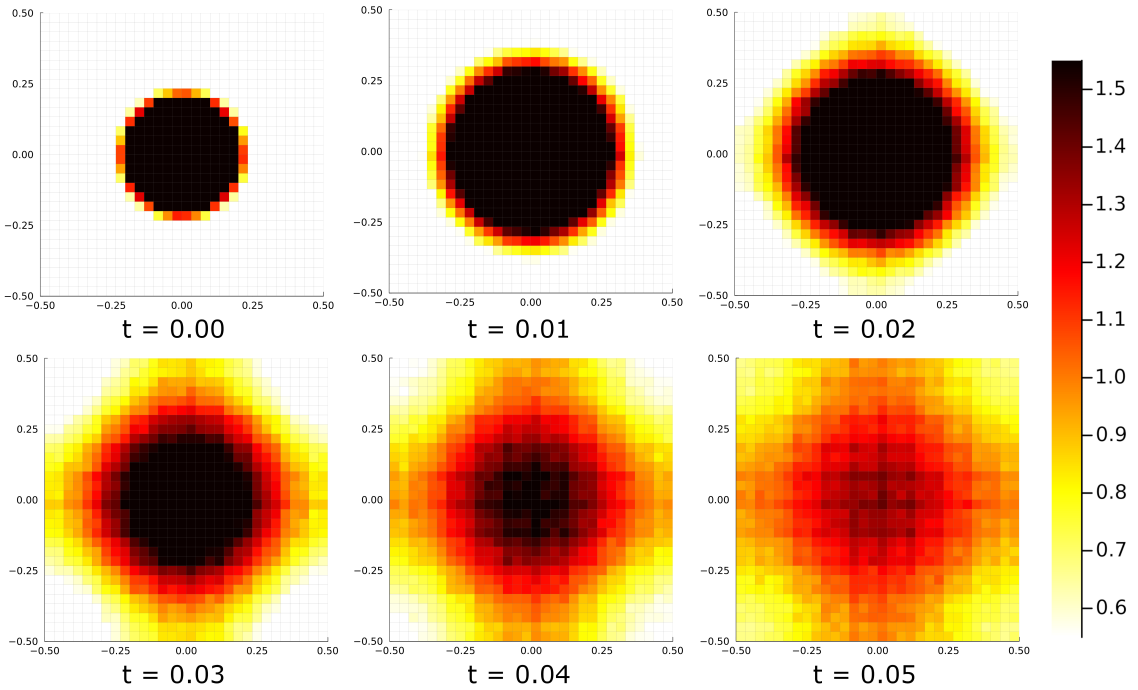

Figure 12: Heatmaps of a Monte Carlo simulation of the hard sphere cell model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. hscmHeatmapscallback30.

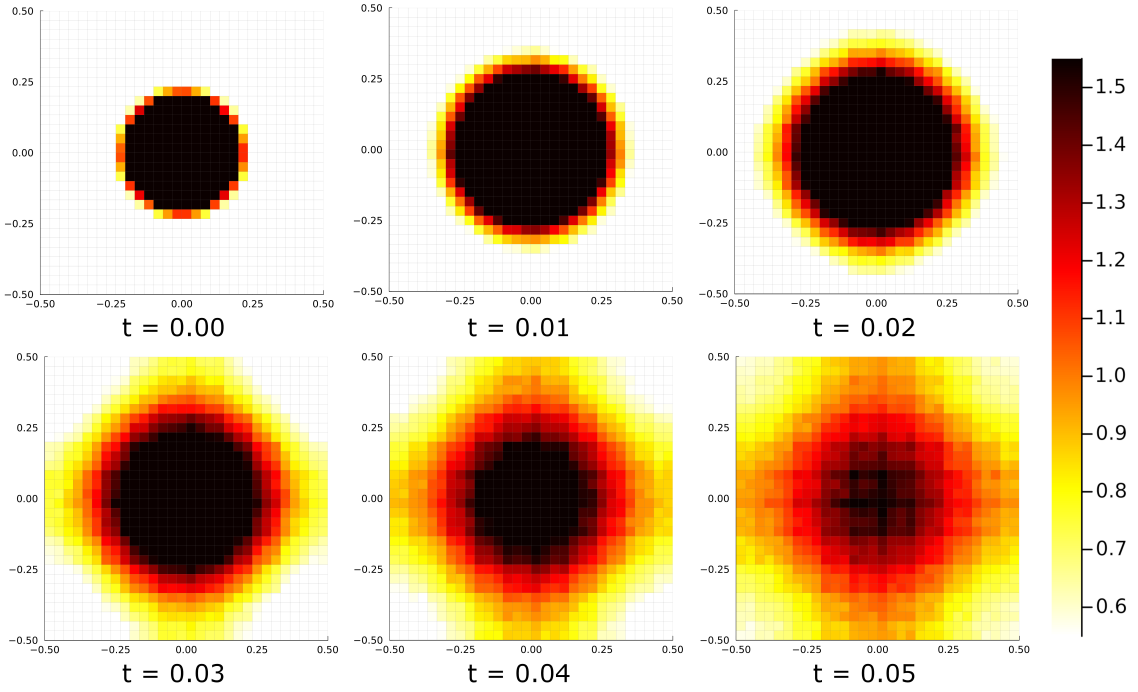

Figure 13: Heatmaps of a Monte Carlo simulation of the point particle model at the times $t \in \{0.00, 0.01, 0.02, 0.03, 0.04, 0.05\}$. They visualise the evolution of the particle density over time. hscmHeatmapsbillard30.

5 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 (*Diffusitivity of 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, July 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.