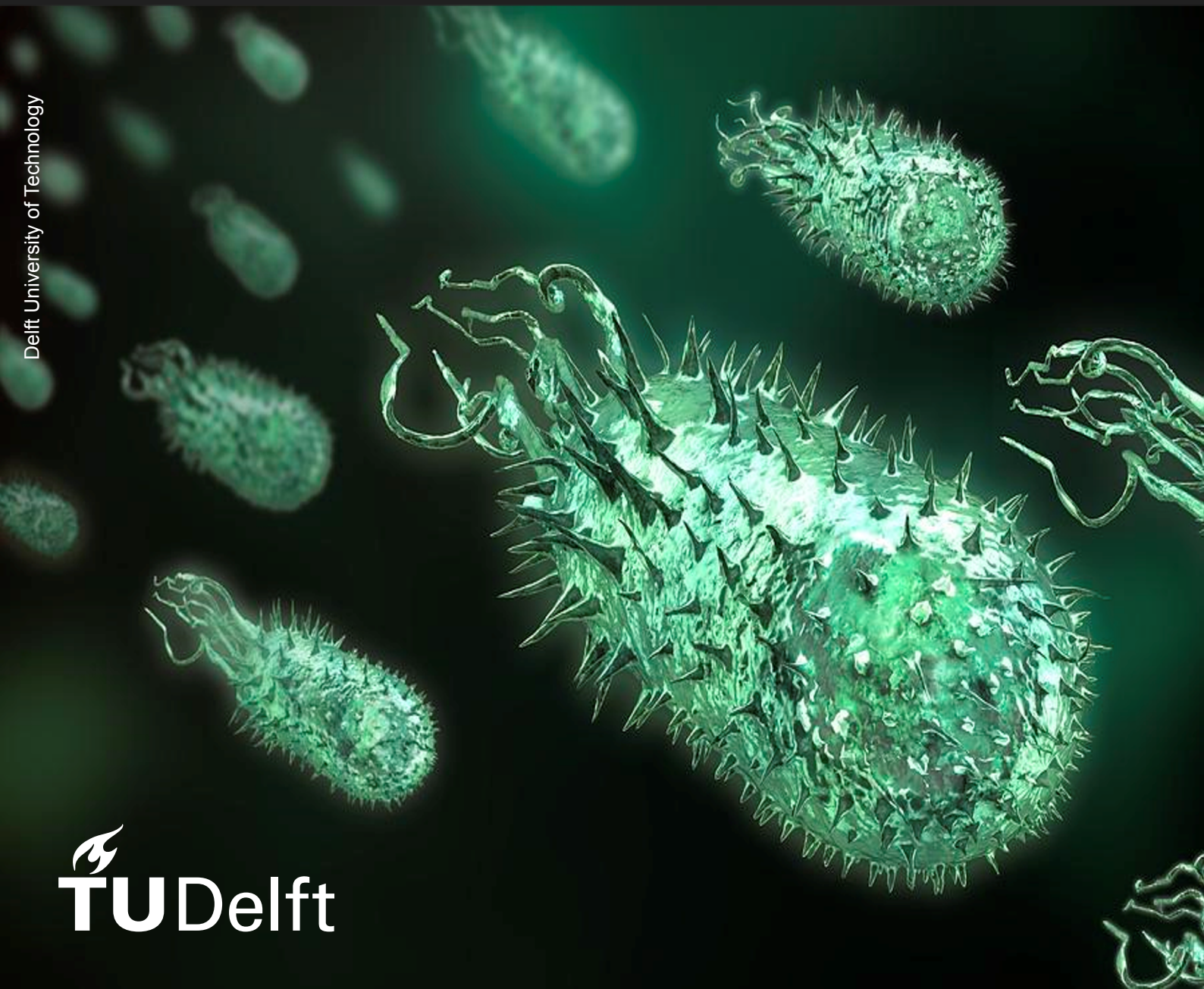
Tiny Chaotic Swimmers Achieving Great Collective Order

A study on the dynamics of self-propelling agents in a bacterial colony

Mees Kersbergen



Delft University of Technology

TU Delft

Tiny Chaotic Swimmers Achieving Great Collective Order

A study on the dynamics of self-propelling agents in a bacterial colony

by

Mees Kersbergen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday May 9, 2022 at 15:00h.

Student number: 4391667
Thesis committee: Dr. T. Idema, TU Delft, supervisor
Dr. J. Zwanikken, TU Delft
Dr. W. Caspers, TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

"From chaos, order can emerge", a counterintuitive statement but also one laying the foundation for complex living systems. Individually acting agents can collectively produce organised structures at a larger scale. Possibly the most ubiquitous example of this phenomenon are bacterial colonies. Everywhere around us, a pandemonium of pushing and pulling produces complex structures. The most researched bacterium is *E. coli* and partially due to its excellent swimming capabilities, the internal structure of its colony is predominately shaped through mechanical repulsion coupled with individual motility. In this thesis, we are interested in the emergent dynamics within a bacterial colony. Our focus will lie on how the colony's density ρ affects the internal structure and motility.

To study the colony's interior, we built a three-dimensional individual-based model (IBM) with self-propelling sphero-cylindrical agents representing *E. coli* bacteria and governed by mechanical interactions. A downside of IBM is its computational costliness, posing an optimisation challenge which will also be covered in this thesis.

A phase transition spontaneously occurs over time from isotropic to an aligned nematic phase. This transition takes longer for higher-density systems. We found a linear relation between the density and the local order for a colony in a quasi-infinite domain. Furthermore, after equilibration, the particles initially behave ballistically. However, this changes to diffusive behaviour in a later stadium. The Reynolds number $Re \propto 10^{-3}$ is two orders of magnitude larger than expected for *E. coli*, possibly due to underestimating the viscosity.

On a final note, the method to determine the moment of equilibrium t_{equi} gives an underestimation in the case of a two-step phase transition; an improved method is proposed.

Contents

1 Introduction

The collective behaviour of active agents is a vibrant topic in the field of biophysics. While agents act individually, complex but organised structures can emerge at a larger scale [1–3]. This phenomenon is found throughout nature, from flocks of birds in the sky to schools of fish in the sea, ranging from immense herds of wildebeest to the smallest colonies of bacteria [4].

In this thesis, we take a deeper dive into collective dynamics at the microscale, into those of bacterial colonies. Bacteria indisputably play a major role in human life; they are so numerous that the number of bacterial cells a single person carries around is similar to that of his or her own cells [5]. Apart from being abundant, bacteria are also useful: they are used in water filtration installations [6]; we use them to make foods like yoghurt; and after production, they assist you in digesting your previous meals in your intestines [7].

Suffice to say, it is advantageous to better understand these creatures. Though there already is a substantial amount of knowledge on the micro and macro scale, research on collective behaviour at a mesoscopic level is still relatively in its infancy [8].

1.1 Theoretical research

The conventional method of studying bacteria is an experimental one, where the focus is typically on the biological and chemical interactions within a colony and its surroundings [9]. However, to study the structural properties of a colony, the mechanical interactions play a role as well [3, 10]; a bacterial colony migrates predominantly due to the reciprocal repulsion force coupled with bacterial growth [11]; the internal structure is then predominately shaped by the same repulsion force coupled with a self-propelling force in case of good swimmers such as *E. coli* [12]. Another important element affecting a colony's structural properties is the extracellular polymeric substance (EPS) on the bacterial cell surface. The EPS makes a bacterium sticky, keeping a colony together, attached to a substrate and constitutes a matrix to form biofilms [13]. Recently, it even became possible to study the structure and the mechanical interactions within a bacterial colony experimentally [14].

Apart from an experimental approach, one can also choose a theoretical one. A theoretical researcher typically does not have pipettes and incubators in his tool belt but instead has an arsenal of formulas, (self-written) software and computer clusters, with which he or she builds models to capture and predict emergent real-world behaviour. By using a theoretical approach, he can explore a broad scope of diverse configurations, both in the sense of testing in larger quantities and in creating more extreme environments, which are not always feasible in a laboratory setting [15, 16].

Theoretical and experimental research can also supplement each other. A theorist can give rise to a new or more specific hypothesis, enabling an experimenter to take a more focused approach [17]. By narrowing the experimental scope, the experimental research becomes less resource-intensive as well; a welcome and beneficial side effect.

Additionally, theoretical research can also fulfil an assisting role post-experiment by enabling the isolation of specific mechanisms in a simulation, which can be a helpful feature when interpreting experimental results.

An example of theoretical research is computational research, a field that has become increasingly prominent in the last few decades due to rapid developments in computing power. In computational research, we impose a particular set of theoretically derived rules onto a system, after which we analyse the emerging behaviour of the system. Together, the experimental and theoretical approach form a delightful synergy to pave the way to a better understanding of these small but ubiquitous creatures.

1.1.1 Two approaches

Within the field of computational research, there are broadly speaking two approaches for modelling bacterial colonies: continuous and discrete. In a continuous model, the bacterial colony is treated as biomass bulk, characterised by macro states like temperature and pressure gradients, and governed

by differential equations. Inherently, a continuous approach is a top-down approach. As a result, a continuous model typically simulates a colony at the macro scale. The benefit of a continuous approach is that it is quantitative and often deterministic [18]. However, including heterogeneity in the model requires a multidimensional formulation, which makes this approach complicated and computationally demanding.

The second approach is a discrete one, which is typically bottom-up: the rules of the system are imposed on discrete elements of the system, where the macro behaviour is emergent from the complex interactions between the elements [19]. The spatial modelling scale in a discrete approach does not have an upper limit in theory. However, in practice, these models typically reside at micro and meso scales since the computational power required increases with the spatial dimensions of the model, often non-linearly. The benefit of this approach is the minimal need for assumptions. Furthermore, it allows for individual variability and individual adaptive behaviour, easing the implementation of heterogeneity into the system [8]. However, this approach introduces randomness into the system, making it difficult to analyse [8, 18]. This approach is in particular useful for describing non-steady state systems with spatial heterogeneity.

Within the discrete approach, models can be subdivided into three categories: cellular automatons (CAs), where the space is divided into a lattice of discrete tiles, the most famous being Conway's game of life [20]; hybrid continuous-discrete models, a variety of a CA model where the mass transport with the surrounding medium, often a substrate, is modelled continuously; and lastly individual-based models (IBMs), where instead of space, the bacteria are discretised and treated as individual agents, by which their spatial orientations are not bound to a lattice [18, 21, 22]. In this thesis, we will dive deeper into the last approach, the individual-based modelling. The main benefits this approach provides are that is requires minimal assumptions and that it enables heterogeneity in the form of individual variability in size and interactions. Therefore, it is well-suited for modelling non-steady state systems with spatial heterogeneity [8]. The drawbacks are its computational costliness and difficulty to analyse.

1.2 Our model

In this thesis, we are interested in the mechanical interactions between self-propelling bacteria and the resulting dynamics of the system. Until now, researchers mainly studied the mechanical interactions due to bacterial growth on a substrate and to a lesser extent due to the bacteria's self-propelling capabilities. To study these effects, we chose to model a bacterium with a sphero-cylindrical shape: the widely researched *E. coli*, since this bacterium is a good swimmer [12]. We chose a theoretical approach, in specific computational research, because it comes which a lot of variability and the ability to perform a large number of controlled experiments. Since we are interested in the mechanical interactions between individual bacteria, it only makes sense to use a bottom-up approach. In order to keep assumptions minimal and enable the examination of the spatial orientation within the bacterial colony, we use an individual-based model (IBM).

Within the Idema research group, there already is a model for simulating bacteria. The model is an individual-based model, where the particles, the bacteria, are represented as rounded rectangles in a 2D environment. The particles exist in a low-Reynolds environment, therefore nullifying any inertial forces. To simulate an infinite volume, the current model places the particles in a box with periodic boundary conditions; particles interact via a repelling force to simulate collision. Additionally, this model is also able to simulate a 3D environment where particles are represented as self-propelling sphero-cylinders. However, no extensive research is performed using this feature as of the start of this thesis project.

1.2.1 Improving the model

However well put together the existing model is, there is always room for improvement and there are even some undesirable aspects that need to be adjusted. The first aspect is drag. The drag is computed as if the particles are circles or spheres, neglecting their elongated shape. This poses a

problem because it does not correspond with real-world physics.

A second aspect does not concern the model itself but rather its implementation, specifically its efficiency. The model was implemented for two-dimensional simulations. However, we are also interested in 3D simulations. The extra dimension does not only add an additional vector component to calculate but also introduces additional neighbouring particles to compute interactions with, hence adding to the computing time. Furthermore, an extra dimension means a larger space to fill, requiring more particles to simulate.

To illustrate this fact, let us make a quick comparison. The additional vector component results in a 1.5 times increase in calculations. The number of neighbouring bacteria to compute interactions with increases as well, with a factor of three[1]. Furthermore, in the existing model in 2D, a simulation would typically contain 1000 bacteria. In order to fill a similar 3D space with the same density, one would need to increase the number of bacteria at least 64 fold[2]. Combining these three factors, we a left with an increase in computation time by a factor of 300, over two orders of magnitude.

In addition to improving some existing aspects, we also want to extend the functionality of the model by including some additional features based on mechanical interactions. In the existing model, the environment is simulated as a box with periodic boundaries to emulate an infinite space. However, in the real world, nothing is infinite, so the inclusion of a solid boundary would be a welcome step closer to reality. We want to adjust the current periodic boundaries so that they can be solidified into repelling surfaces; introducing the environment either as an infinite space, two infinite plates, an infinite square tube, or as a solid box. Moreover, we want to incorporate the option for a solid cylindrical boundary, emulating a narrow vessel. Based on Stork et al. [13], we want to implement an adhesive interaction for both the particles reciprocally as well as between the particles and the boundaries. This would emulate the adhesive effects of the extracellular polymeric substances (EPS) attached to a bacterium. The last new feature to implement is a friction force between the particles and the solid boundaries to simulate surface roughness.

Together, the adjusted drag force, supplementary mechanically driven interactions and broadened range of environments should orchestrate a more realistic and comprehensive model. Decreasing the computation time significantly using computational optimisation is a necessity and would enable us to simulate and analyse spatial orientations in motile 3D colonies. To set a reasonable aim, we look at Hartmann et al. [14], who are able to analyse up to 10 000 bacteria in a single biofilm. This number would be a nice aim for our computational optimisation as well.

1.3 Research goal

This thesis aims to create a better understanding of the internal structure of bacterial colonies. Typically, bacterial research is focused on the growing bacteria, while the internal structure is mainly governed by the self-propelling of bacteria. Therefore, we concentrate on this self-propelling nature and the emergent dynamics.

Based on an existing model [23], we want to design a 3D model of a motile bacterial colony with a physically accurate drag, with both solid and periodic boundaries in multiple geometries, and additional interactive forces such as adhesion and friction. Furthermore, to satisfy the need for a larger number of particles, we want to significantly improve the performance of the computational implementation, setting a goal of 10 000 simulated particles.

Once we have an extended model with improved drag and a more efficient implementation, we can have a closer look at the emerging dynamics in the system. We are specifically interested in how the density of a motile bacterial colony affects the dynamics of the system. Subsequently, we are interested if this behaviour differs for the different geometries our system will have to offer. Furthermore, we want to investigate how different interaction strengths for the proposed adhesive interaction affect the dynamics of our bacterial colony.

[1] Here a dense smectic packing is assumed, increasing the number of neighbouring bacteria from $9 - 1 = 8$ in 2D to $27 - 1 = 26$ in 3D, resulting in a factor $26/8 \approx 3$

[2] Again the assumption for a dense smectic packing is made, as well as a length-to-width ratio of 4; resulting in a increase of $4\sqrt{1000/4} \approx 64$ the number of bacteria.

2 Theory

In this chapter, we will discuss the physics behind the system we built. We start with a particle representing a single bacterium, after which we continue by introducing its interactions, both between particles as well as between a particle and the environment. After which, we introduce the different geometries of possible simulation environments. We finalise this chapter with an explanation on natural units and how our system relates to the physical world.

2.1 Single bacterium

The *E. coli* bacteria are of the kind *bacilli*, meaning they are elongated in shape with round edges. In our model, a bacterium is therefore represented as a sphero-cylindrical particle. This sphero-cylinder consists of two spheres with centre points A and B. The spheres have a radius R and are rigidly connected over a distance L_{inner}, resulting in a total length

$$L = L_{\text{inner}} + 2R. \tag{1}$$

Furthermore, we define the aspect ratio a of a particle as

$$a = \frac{L}{2R}. \tag{2}$$

In the actual simulation, we set the a and R, and calculate the corresponding lengths L and L_{inner} using eq. 1 and 2. Lastly, we introduce a polarity by defining a direction $\hat{\mathbf{u}}$, using the positions of points A and B, respectively $\mathbf{x_A}$ and $\mathbf{x_B}$,

$$\hat{\mathbf{u}} = \frac{\mathbf{x_B} - \mathbf{x_A}}{\|\mathbf{x_B} - \mathbf{x_A}\|}. \tag{3}$$

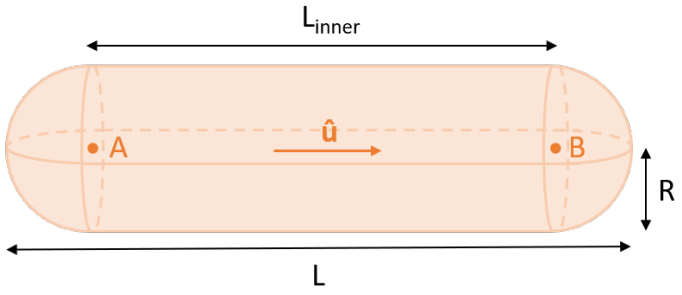


Figure 1: Schematic illustration of a single particle with length L, inner length L_{inner}, radius R, direction $\hat{\mathbf{u}}$, and acting points A and B, where the last two determine the position and orientation of the particle.

Now that we have defined the form factor of the particle, we can introduce forces. All forces acting on a particle are distributed over the centre points A and B of the two spheres. In the rest of the thesis, we will simply refer to the spheres of the particle. First of all, because the distinction of the points A and B is arbitrary for the forces acting on them. Secondly, because all forces, velocities and positions are considered for both spheres individually, making the consideration of the particle as a whole unsuitable.

The first force we will introduce is the self-propulsion force $\mathbf{F_{sp}}$, which simulates the swimming of bacteria and consequently facilitates a dynamic system. The force is equally distributed over the two spheres and defined as

$$\mathbf{F_{sp}} = f_{\text{sp}}\hat{\mathbf{u}}, \tag{4}$$

where f_{sp} is the self-propulsion constant.

2.2 Interactions

Apart from the non-interactive self-propulsion force, we can consider three types of interactive forces: repulsive forces, adhesive forces[3] and friction forces. We can place these types in at least one of two categories: particle-particle (pp) interactions and particle-surface (ps) interactions. We will elaborate upon these surfaces later in section 2.4.

Table 1: The types of interactions in our model.

Interaction	particle-particle	particle-surface
repulsion	x	x
friction		x
adhesion[3]	x	x

2.2.1 Repulsion

The most fundamental interaction is repulsion, which we use to simulate collisions between both particles and surfaces, thus preventing them to move through each other. Similar to Storck et al. [13], we model the repulsion as a spring force, which scales linearly with the overlap between two bodies

$$\mathbf{F_{ov}} = -k_{\mathrm{ov}}\boldsymbol{\delta}. \tag{5}$$

Here, $\mathbf{F_{ov}}$ is the overlap force, which enables the repulsion, k_{ov} is the spring constant, and $\boldsymbol{\delta}$ is overlap vector, such that its magnitude is the amount of overlap between two bodies. This overlap vector $\boldsymbol{\delta}$ is calculated using the "shortest distance between lines"-method [24], which returns the shortest distance vector $\mathbf{d}_{ij}$, between bodies i and j. Depending on the interaction, we use

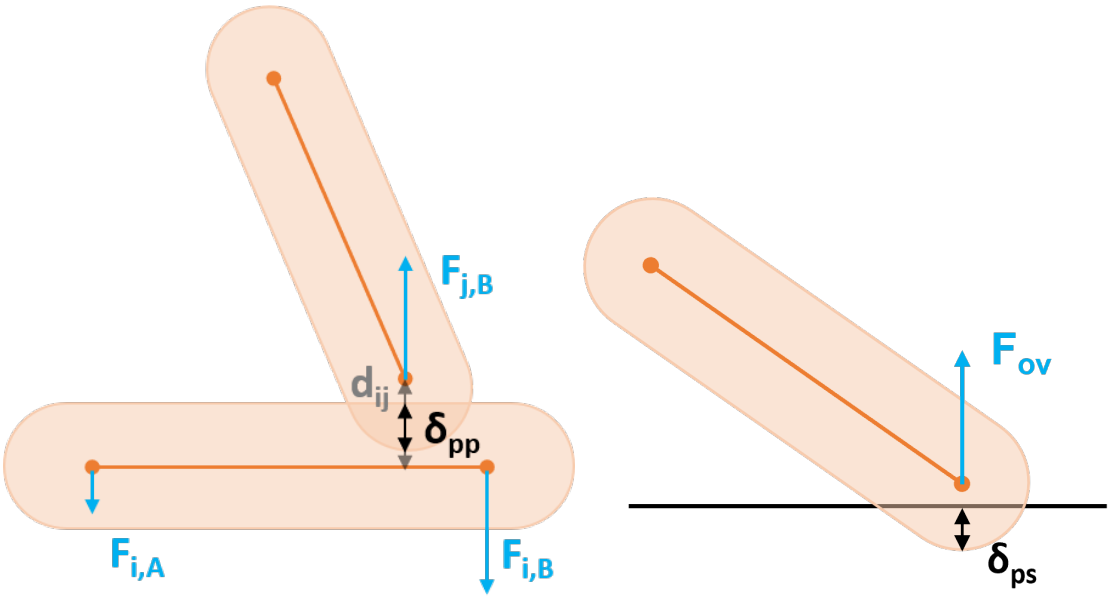
$$\boldsymbol{\delta_{pp}} = (2R - \|\mathbf{d}_{ij}\|)\hat{\mathbf{d}}_{ij}, \tag{6}$$
$$\boldsymbol{\delta_{ps}} = \ (R - \|\mathbf{d}_{ij}\|)\hat{\mathbf{d}}_{ij}, \tag{7}$$

for either the particle-particle interaction, or particle-surface interaction respectively, where R is the particle radius.

For particle-particle collisions, it is not immediately clear how to distribute the forces over the two spheres A and B. For this case, we distribute the forces depending on the relative point of collision on the spine of the particle, indicated with coefficient $s \in [0, 1] \cap \mathbb{R}$:

$$\mathbf{F_{A,ov}} = s\mathbf{F_{ov}}, \tag{8}$$
$$\mathbf{F_{B,ov}} = (1 - s)\mathbf{F_{ov}}. \tag{9}$$

[3]The adhesive interaction can be toggled on/off and is therefore not always present. In addition, it has a different implementation for particles and surfaces.



(a) The distribution of overlap forces F over spheres A and B as a result of a collision between particle i and j, depending on the overlap vector $\boldsymbol{\delta_{pp}}$.

(b) The overlap force $\mathbf{F_{ov}}$ as a result of a collision with the surface, depending on the overlap vector $\boldsymbol{\delta_{ps}}$ between a particle and the surface

Figure 2: The overlap force (in blue) between two particles and between a particle and a surface, depending on overlap vector $\boldsymbol{\delta}$.

2.2.2 Friction

The second interaction exists only between particles and surfaces. The friction force $\mathbf{F_f}$ inhibits movement alongside the surface, simulating the roughness of the surface.

To determine the direction of $\mathbf{F_f}$ at time step t, we first introduce vector $\mathbf{q}$, defined as the parallel projection of the sphere velocity on the surface (eq. 10). The velocity $\mathbf{v_{t-1}}$ is the velocity of the sphere at the surface at the previous time step $t-1$. The direction of $\mathbf{F_f}$ then opposite to $\mathbf{q}$. The magnitude of $\mathbf{F_f}$ is proportional to the friction coefficient μ and the magnitude of overlap force $\mathbf{F_{ov,ps}}$ between the particle and the surface 11. Lastly, $\mathbf{F_f}$ is bound by the self-propulsion force $\mathbf{F_{sp}}$ through hard coding, resulting in the following set of equations:

$$\mathbf{q} = \mathrm{proj}^{\parallel}_{\mathrm{surf}}(\mathbf{v}_{\mathrm{t-1}}), \tag{10}$$

$$\mathbf{F}_{\mathrm{f}} = -\mu \|\mathbf{F}_{\mathrm{ov,ps}}\| \hat{\mathbf{q}}, \tag{11}$$

$$\|\mathbf{F}_{\mathrm{f}}\| \leq \|\mathbf{F}_{\mathrm{sp}}\|. \tag{12}$$

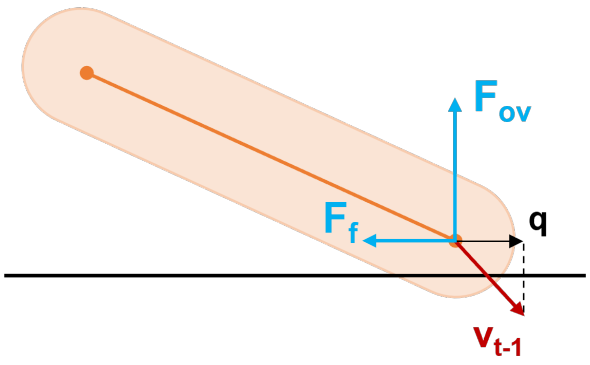


Figure 3: The determination of friction force $\mathbf{F_f}$ as a result of the overlap force $\mathbf{F_{ov}}$, where the direction is opposite to the parallel projection $\mathbf{q}$ of the velocity $\mathbf{v_{t-1}}$ at the previous time step.

2.2.3 Adhesion

The last interaction has two implementations, one between particle and surface, and one between particles, and can be toggled on or off in our simulation. Similar to the repulsive force, we model the adhesion as a spring force; an approach based on Storck et al. [13]. Furthermore, similar to the friction, an adhesive interaction is assessed per particle sphere.

Let us start by discussing the particle-surface adhesive interaction. To assess the existence of an adhesive spring, we consider two lengths: the distance L_{form} over which an adhesive spring interaction is formed, and the maximum distance L_{break} over which an adhesive spring interaction can exist, where $L_{\text{form}} < L_{\text{break}}$, see Figure 4. If, for either of the spheres, the distance to the surface distance $dx_{\odot,s}$ is less or equal to L_{form}, then a spring between that sphere and the surface is formed, resulting in the adhesive spring force

$$\mathbf{F}_{\text{adh,ps}} = k_{\text{adh}}(dx_{\odot,\text{s}} - R)\hat{\mathbf{n}}, \tag{13}$$

where k_{adh} is the adhesive spring constant, and $\hat{\mathbf{n}}$ is unit vector orthogonal to the surface, pointing away from the centre of the simulation environment. As seen from eq. 13, the rest length of the spring is equal to the particle radius R. Once a spring is formed, it exists until $dx_{\odot,\text{s}}$ surpasses the maximum spring length L_{break}, where the spring breaks. Apart from the two possible sphere-surface springs, there is a third spring that can be formed, between the particle centre and the surface. The spring existence and force is assessed using the particle centre-to-surface distance $dx_{\odot,\text{s}}$, and governed by the same set of rules, with the exception that the resulting force is equally distributed over the two spheres.

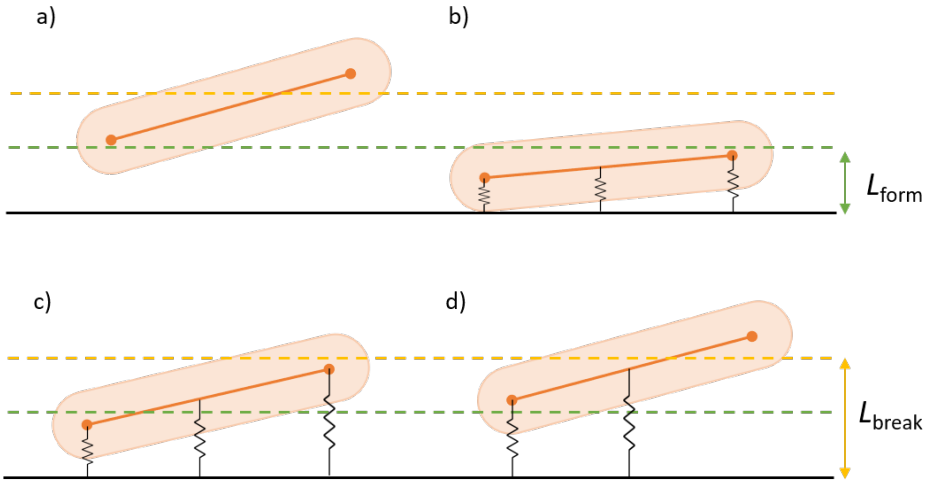


Figure 4: Different scenarios for when an adhesive spring between a particle and the surface will form and when it will break, dependent on form length L_{form} and break length L_{break}.

In addition to the particle-surface adhesion, we consider particle-particle adhesion. This is based on a similar principle, with spring connections between the particle spheres, but now we extend this model by introducing additional crossing springs. This idea is adopted from Storck et al. [13] and forces smectic alignment on the particles, something we also observe in actual bacterial colonies (Dr. Idema, 2021). Applying this method results in a maximum total of four acting spring forces described by

$$\mathbf{F}_{\text{adh,pp}} = k_{\text{adh}}(dx_{\odot,\odot} - L_{\text{rest}})\hat{\mathbf{dx}}, \tag{14}$$

where $dx_{\odot,\odot}$ is the sphere-to-sphere distance, L_{rest} is the length of the spring and $\hat{\mathbf{dx}}$ the direction of the sphere-to-sphere spring connection. The spring rest length L_{rest} is such that there are no acting forces when the particles are in perfect smectic alignment (see Figure 5e):

$$L_{\text{rest}} = \begin{cases} \sqrt{4R^2 + L_{\text{inner}}^2}, & \text{for diagonally crossing springs} \\ 2R, & \text{other.} \end{cases} \tag{15}$$

Our first condition for the presence of an adhesive interaction, is vicinity, ensured by the break and formation lengths, L_{form} and L_{break}, of the springs. The break and formation lengths for diagonally crossing springs are defined using the Pythagoras' theorem as well

$$L_{\text{X}} = \begin{cases} \sqrt{L_{\text{X,default}}^2 + L_{\text{inner}}^2}, & \text{for diagonally crossing springs} \\ L_{\text{X,default}}, & \text{other} \end{cases} \tag{16}$$

where L_X is a substitute for either L_{form} or L_{break}.

The second condition for the formation of springs is a small-angle condition, where we specify an upper limit on the inner product between the directions of two particles

$$\hat{\mathbf{u}}_i \cdot \hat{\mathbf{u}}_j \leq \cos\theta_{\text{limit}}, \tag{17}$$

characterised by the maximum angle θ_{limit}. This condition ensures a large sticking area between the particles and thus prevents perpendicular particles with little exposed sticking area to form adhesive links (see Figure 5a).

The last condition for the spring formation is that if two particles can form adhesive springs, there should be at least three (see Figure 5b). This way, we prevent adhesion springs from applying a pushing force between spheres.

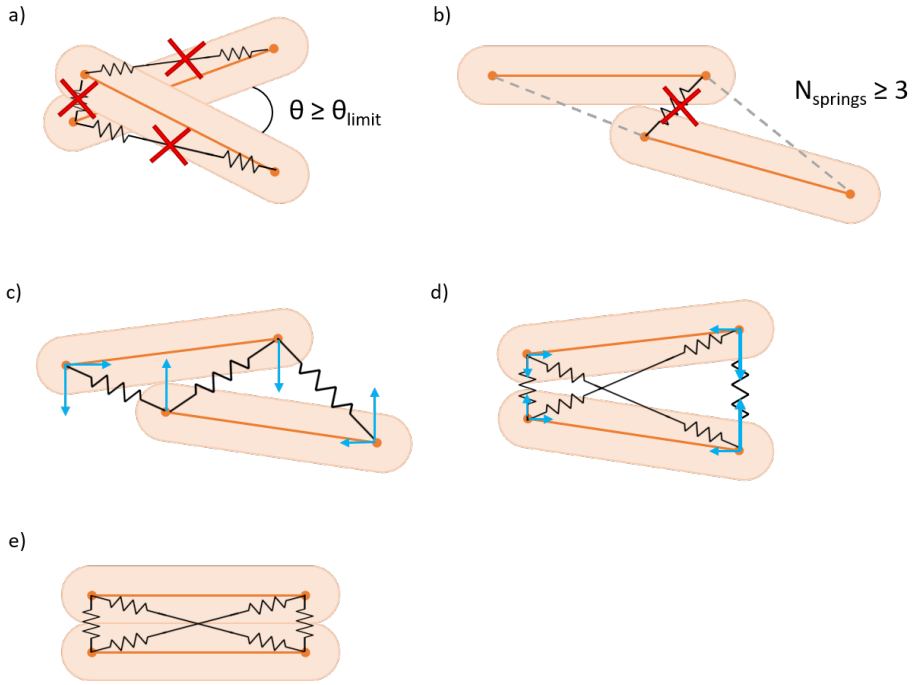


Figure 5: Particle-particle adhesion spring formation with acting forces. In a) the angle θ between two particles is larger than the critical angle θ_{limit}, so no springs are formed. In b) only two ends are close enough to form a spring; therefore no springs are formed since the requirements for at least three springs are not met. In c) three pairs of ends are close enough to form three springs, resulting in four acting forces (in blue). In d) an additions fourth spring is formed since even the furthest two ends are close enough, resulting in four acting forces (in blue). In e) there are four springs without any acting forces since the particles are as close as possible, resulting in all springs being at their rest lengths.

2.3 Equation of motion & drag force

Now that we have our self-propulsion force $\mathbf{F_{sp}}$, both overlap forces $\mathbf{F_{ov,\,pp}}$ and $\mathbf{F_{ov,\,ps}}$, friction force $\mathbf{F_f}$ with the surface, and both adhesion forces $\mathbf{F_{adh,\,pp}}$ and $\mathbf{F_{adh,\,ps}}$, we can build our equations of motion. For that, we will need to incorporate one last force: the drag force $\mathbf{F_D}$, which is result of swimming through the surrounding medium.

Using Newton's second law, we can get to an equation of motion. Fortunately, our system becomes quite forgiving in complexity because bacteria typically live in a low Reynolds environment [14, 25]. Therefore, we can neglect acceleration, resulting in the following equation of motion:

$$0 = \mathbf{F_{sp}} + \mathbf{F_{ov,\,pp}} + \mathbf{F_{ov,\,ps}} + \mathbf{F_{adh,\,pp}} + \mathbf{F_{adh,\,ps}} + \mathbf{F_f} - \mathbf{F_D}. \tag{18}$$

To get from this equation to an expression for the actual motion of the particle, we can use the drag force $\mathbf{F_D}$. In a low Reynolds regime, we recognise this drag as Stokes' drag, resulting in a drag force that relates linearly to the velocity $\mathbf{v}$. Subsequently, we can multiply this velocity with a small time step Δt to get the positions of the particle at every discrete moment in time.

Now that we have the road map, we need to walk it. The first step is to rewrite the equation of motion such that the drag force $\mathbf{F_D}$ equals the sum of all other forces

$$\mathbf{F_D} = \sum \mathbf{F}. \tag{19}$$

As mentioned previously, our next step is to substitute $\mathbf{F_D}$ for the velocity $\mathbf{v}$, together with a drag coefficient γ. However, we cannot simply equal this to the sum of the forces. An important aspect to consider is that all forces act on the individual spheres instead of the entire particle. Therefore, we need to consider them separately as the net forces $\mathbf{F_A}$ and $\mathbf{F_B}$:

$$\gamma \mathbf{v} = \mathbf{F_A} + \mathbf{F_B}. \tag{20}$$

For the value of the drag coefficient γ, we use the numerically derived drag coefficients for cylinders by Tirado et al. [26], which are demonstrated to be valid for sphero-cylinders as well by Bartuschat et al. [25]. The drag coefficients distinguish parallel, perpendicular and rotational motion respectively as

$$\gamma_{\parallel} = \frac{2\pi\eta L}{\ln(a) - 0.207 + 0.980/a - 0.133/a^2}, \tag{21}$$

$$\gamma_{\perp} = \frac{4\pi\eta L}{\ln(a) + 0.839 + 0.185/a + 0.233/a^2}, \tag{22}$$

$$\gamma_r = \frac{1}{3}\frac{\pi\eta L^3}{\ln(a) - 0.662 + 0.917/a - 0.050/a^2}, \tag{23}$$

where a is the aspect ratio as mentioned in section 2.1. These equations are valid in the regime $2 < a < 30$, which is adequate for our system since *E. coli* have a typical aspect ratio between 2 and 4 [27].

The next obstacle on our road is transforming the forces $\mathbf{F_A}$ and $\mathbf{F_B}$ to parallel, orthogonal and angular components to correspond to Tirado's drag coefficients which consider the particle in its entirety. Respectively, the parallel, orthogonal and angular components are given by

$$\gamma_{\parallel}\mathbf{v}_{\parallel} = \tfrac{1}{2}\big(\mathbf{F}_{A,\parallel} + \mathbf{F}_{B,\parallel}\big), \tag{24}$$

$$\gamma_{\perp}\mathbf{v}_{\perp} = \tfrac{1}{2}(\mathbf{F}_{A,\perp} + \mathbf{F}_{B,\perp}), \tag{25}$$

$$\gamma_r \boldsymbol{\omega} = \tfrac{L_{\text{inner}}}{2}\hat{u} \times \tfrac{1}{2}(\mathbf{F}_A - \mathbf{F}_B). \tag{26}$$

Let us first consider the parallel and orthogonal velocities. For the parallel component we can use an orthogonal projection on the particle direction $\text{proj}_{\hat{u}}$ [4]. For the orthogonal component, we need to introduce unit vector $\hat{v}$ perpendicular to the particle direction $\hat{u}$, pointing radially outward. Because the particle will move in the same direction as the net force, we choose the direction of $\hat{v}$ accordingly

$$\hat{v} \parallel \sum \mathbf{F} - \text{proj}_{\hat{u}}\Big(\sum \mathbf{F}\Big). \tag{27}$$

Equipped with the parallel and orthogonal vector components, we can write the corresponding velocities as

$$\mathbf{v}_{\parallel} = \frac{1}{\gamma_{\parallel}} \text{proj}_{\hat{u}}\left(\frac{\mathbf{F}_A + \mathbf{F}_B}{2}\right), \tag{28}$$

$$\mathbf{v}_{\perp} = \frac{1}{\gamma_{\perp}} \text{proj}_{\hat{v}}\left(\frac{\mathbf{F}_A + \mathbf{F}_B}{2}\right). \tag{29}$$

For the rotational component of the velocity, we initially only consider its magnitude, i.e. the angular speed

$$\omega = \frac{L_{\text{inner}}}{2\gamma_r} \left\| \text{proj}_{\hat{v}}\left(\frac{\mathbf{F}_A - \mathbf{F}_B}{2}\right) \right\|. \tag{30}$$

The expression for the translational displacement $\mathbf{\Delta x_t}$ is quite straightforward derived from the velocities. Subsequently, we can define the displacement due to rotation $\mathbf{\Delta x_r}$ using the angular speed ω combined with a direction expressed in terms of $\hat{u}$ and $\hat{v}$ using trigonometric functions. To simplify our calculations, we use a small-angle approximation to get our approximated rotational displacement[5] $\mathbf{\Delta x'_r}$, which is valid due to very small angular speeds. This leaves us with the following expressions

$$\mathbf{\Delta x_t} = \big(\mathbf{v}_{\parallel} + \mathbf{v}_{\perp}\big)\Delta t, \tag{31}$$

$$\mathbf{\Delta x_r} = \frac{L_{\text{inner}}}{2}\Big(\Big[1 - \cos(\omega \Delta t)\Big]\hat{u} + \sin(\omega \Delta t)\,\hat{v}\Big), \tag{32}$$

$$\mathbf{\Delta x'_r} = \frac{L_{\text{inner}}}{2}\omega \Delta t\,\hat{v}. \tag{33}$$

At last, we conclude our journey with the arrival at an expression for the displacement for every sphere. A final note is that the angular displacement is opposite for the two different spheres, resulting in a final expression for the displacements

$$\mathbf{\Delta x_A} = \mathbf{\Delta x_t} + \mathbf{\Delta x'_r}, \tag{34}$$

$$\mathbf{\Delta x_B} = \mathbf{\Delta x_t} - \mathbf{\Delta x'_r}. \tag{35}$$

However, since this induces a small length increase, which accumulates over time, we rescale the length around the centre, see Appendix A.2 for an exemplification of this phenomenon.

[4] $\text{proj}_{\hat{u}}(\mathbf{y}) \equiv \frac{\hat{u}\cdot\mathbf{y}}{\hat{u}\cdot\hat{u}}\hat{u} = (\hat{u} \cdot \mathbf{y})\hat{u}$

[5] An additional term $\frac{L_{\text{inner}}}{2}\omega_2 \Delta t\,\hat{w}$ should be added to the rotational displacement $\mathbf{\Delta x'_r}$, with direction $\hat{w} = \hat{u} \times \hat{v}$ and angular speed $\omega_2 = \frac{L_{\text{inner}}}{2}\left\|\text{proj}_{\hat{w}}\left(\frac{\mathbf{F}_A - \mathbf{F}_B}{2}\right)\right\|$, see section 5.1.3 for an explanation.

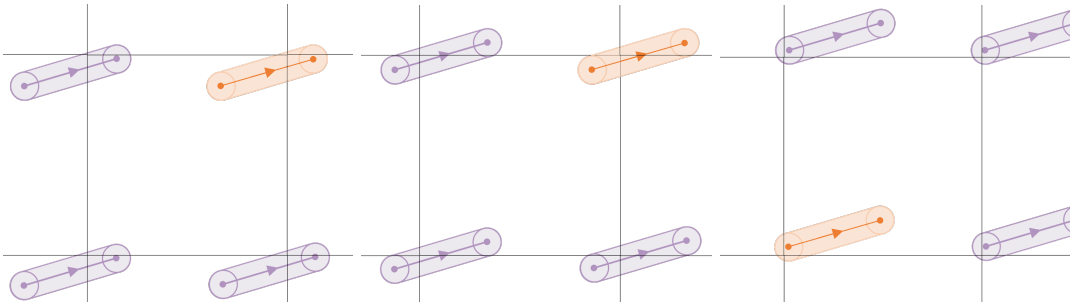
2.4 Environments

At the start of this project, the model was able to simulate a periodic box in either 2D or 3D. In this section, we will describe the addition of solid boundaries as well as a circular in 2D and a tubular geometry in 3D. Furthermore, some interactions with these boundaries like repulsion, friction and adhesion are added, as presented in section 2.2.

To emulate infinite space, we built a cubic container with periodic boundaries, ranging from $-S$ to S. This spatial parameter S is determined by the number of particles N and the density ρ of the system, where the density is defined as the fraction of space the particles cumulative occupy over the total space of the environment. For a 2D box simulation this would be the cumulative particle area A_p over the total area. For a 3D cube simulation, the density would be the cumulative particle volume V_p over the total volume, resulting in a spatial parameter

$$S = \frac{1}{2}\sqrt{\frac{NA_p}{\rho}}, \quad \text{in 2D, and} \qquad S = \frac{1}{2}\sqrt[3]{\frac{NV_p}{\rho}}, \quad \text{in 3D.} \tag{36}$$

If a particle is near the periodic boundary, then a translated copy is created on the other side as well, referred to as a ghost particle, as illustrated in a 2D example in Figure 6. Only after the centres of both spherical ends have crossed the boundary, the original particle is translated to the other side of the periodic box. The Verlet list, which we will discuss in section 3.1.2, takes this periodicity into account by checking the proximity to all ghosts, for all particles near the periodic boundary.



(a) Particle (orange) with its three ghosts (purple), moving right upwards.

(b) Particle (orange) with its three ghosts (purple), moving right upwards.

(c) The second point of the particle (orange) just crossed the boundary, relocating the particle to the other side of the box.

Figure 6: Schematic representation of a particle (orange) with three ghost particles (purple) in a 2D box environment with periodic boundaries.

In addition, every pair of opposite facing boundaries can be solidified into surfaces. Solidifying a boundary means enabling the repulsive and friction interaction, and optionally the adhesive interaction. This facilitates six additional environments: two infinite plates, an infinitely long square tube, and lastly, a finite cubic box with sides of length $2S$, all either with or without adhesive interaction.

Furthermore, we implemented an infinite tube environment as a cylinder with periodic ends on both flat sides. Like with the box, we can set the number of particles N, whereafter we calculate the spatial parameter

$$S = \sqrt[3]{\frac{NV_p}{2\pi\rho a_{tube}}}, \tag{37}$$

which in this case represents the tube radius. Here, a_{tube} is the aspect ratio of the tube, i.e. the length over the diameter.

Alternatively, specific to the tube environment, we can also specify a tube radius S, instead of the number of particle N. In this case, the program determines an integer value for N based on the given

S, after which the tube radius S is adjusted according to eq. 37, to correspond with this integer number N. This method ensures an integer number of particles while nearly matching the set tube radius.

2.5 Natural units and parameter values

To streamline our calculations, we express our simulation in natural units. Using natural units, we choose certain quantities to be equal to one[6], see eq. (38-40). Here, the natural quantity is depicted with a tilde. To convert these back to SI units we use conversion factors, depicted with a knot, based on real-world values. In our simulation, we chose the following quantities to equal one:

$$\tilde{R} = \frac{R}{R_0} = 1, \qquad \text{with} \quad R_0 = 0.5\mu\text{m}, \tag{38}$$

$$\tilde{k} = \frac{k}{k_0} = 1, \qquad \text{with} \quad k_0 = 0.01\text{N/m}, \tag{39}$$

$$\tilde{\gamma} = \frac{\gamma}{6\pi\eta R_0} = \frac{\tilde{L}}{3f(a)}, \quad \text{with} \quad \eta = 1.0016 \cdot 10^{-3}\text{Pa s} \tag{40}$$

with the conversion factors from literature, respectively [29–31].

After normalising certain variables, we can express the rest of the system variables using the conversion factors from eq. (38-40), as shown in Table 2. Here, we also present the values of certain variables in the system, expressed natural units.

Table 2: List of variables with their chosen values in the simulation, which can be multiplied by the conversion unit to get the real-world value in SI units.

	Quantity	**Simulation value**	**Conversion unit**
Normalised variables			
	R	1	R_0
	k_{ov}	1	k_0
	γ	$\frac{L}{3f(a)}$	$6\pi\eta R_0$
System variables			
	a	4	-
	μ	0.01	-
	f_{sp}	0.05	$k_0 R_0$
	$k_{\text{ov,ps}}$	1	k_0
Adhesion variables			
	$k_{\text{adh,pp}}$	0.0005	k_0
	$k_{\text{adh,ps}}$	0.0005	k_0
	$L_{\text{rest,adh,pp}}$	2	R_0
	$L_{\text{form,adh,ps}}$	1.2	R_0
	$L_{\text{break,adh,ps}}$	1.5	R_0

2.5.1 Time scale

Starting with the equation of motion, we can derive a time scale for our system. The dominant force in our equation of motion eq. 18 is the overlap force F_{ov}, which is orders of magnitude larger than all other forces, enabling us to neglect the other components, leaving us with a simplified expression

[6]To comply with the bacterial growth model developed in the same group [28], we chose the simulation value such that the conversion unit would equal that of the other model. Hence the non-normalised drag coefficient γ, with drag constant of a sphere $6\pi\eta R_0$ as conversion unit.

$$F_D = F_{\text{ov}}, \tag{41}$$
$$\gamma_{\parallel}\dot{x} = -2k_{\text{ov}}x. \tag{42}$$

Here, we want to extract the characteristic time by assuming the shortest reaction times. Therefore, we chose the smallest of our drag coefficients, the parallel drag coefficient $\gamma_{\parallel}$. Furthermore, we assume two bacteria moving towards each other, hence introducing the factor 2 in eq. 42. This ordinary differential equation can easily be solved resulting in

$$x = e^{-t/\tau}, \quad \text{with} \quad \tau = \frac{\gamma_{\parallel}}{2k_{\text{ov}}}, \tag{43}$$

where τ is the characteristic time of our system. To make sure we do not overshoot, we perform our calculations with time steps $\Delta t = 0.2\tau$. We save our progress every 300 time steps making a single saved time step $t_{\text{save}} = 300\Delta t$. When we substitute these, including $\gamma_{\parallel}$ from eq. 21, we get

$$t_{\text{save}} = \frac{120a\pi\eta R_0}{k_0 f(a)}, \tag{44}$$

where $f(a)$ is a function of the aspect ratio a, denoting the denominator from eq. 21. The particles in our system always have an aspect ratio of 4, resulting in a saved time step of $t_{\text{save}} = 53$us and a simulated time of 0.42s for a typical run of 8000 saved time steps.

3 Computation

For this project, we wrote a program able to build both 2D and 3D simulation of bacterial colonies, as already introduced in section 1.2. In this chapter, we will discuss the computer science side of the simulation.

In the simulation, we can distinguish two phases. The first phase is the data generation phase, where we run an executable from compiled C++ code in parallel on a computer cluster. Here the program calculates the positions of all particles for a given number of time steps, where every 300 time steps the positions are saved to save memory while maintaining accuracy. We save the data to a JSON file, using the nlohmann library [32]. This file format is chosen for easy usage and general applicability between different platforms and in different programming languages. The second phase is the data analysis and animation phase, where we visualise the simulated particles and analyse the emerged behaviour and properties of the system in Python.

For performance reasons, we write our application in C++17; a programming language notorious for its speed and efficient memory usage [33]. We compile using CMake 3.15 and WSL (Windows Subsystem Linux) Ubuntu 18.04 from Microsoft to cross-compile on Windows for the Linux OS on which the cluster runs.

3.1 Optimisation

We started with a 2D system, which was then extended to a more general system which enabled 3D simulations. However, to overcome small-system size effects, we need to increase the number of particles. Furthermore, the additional vector component in 3D leads to an increase in computations. This drastically increased the computation time, by over two orders of magnitude, as discussed in the introduction (section 1.2.1. In conclusion, there is an urgent need for a performance increase.

3.1.1 Parallel computing

Since the cluster consists of 960 processing cores, a logical step to fully utilise the capability of the cluster is to write the application such that it runs in parallel. For this we use the OpenMP library, which can distribute a workload over multiple threads with shared memory. In our code, we instruct the program to take the maximum number of threads available, which we typically set to 20 in the shell script we use to execute the program.

Because we use CMake, we have to add the appropriate flags in the CMakeList file. Hereafter, we can add the parallelisation with a single line of code above the relevant for-loops. Here, we must specify which information is shared between processing cores and which is private to a single core to prevent memory leaks.

3.1.2 Verlet list method

The main problem of the simulation was scaling. By checking the interaction between every particle, for N particles, the system had a complexity of $\mathcal{O}(N^2)$. To improve on this, we introduce the Verlet list method [34]: for every particle, we create a list in which we keep track of the neighbouring particles and check only for interactions between a particle and its n neighbours, which reduces the complexity to $\mathcal{O}(Nn)$, where theoretically $n \leq N$, but typically $n \ll N$. However, because we have a dynamic system, the number of particles that are considered to be neighbours change over time, requiring the Verlet lists to be updated over time. The process of updating the Verlet lists has a complexity of $\mathcal{O}(N^2)$. So ideally, we do not update too often.

To determine which particles are neighbours and therefore need to be included in our Verlet list, we introduce two different radii; the interaction radius R_{int} and the skin radius R_{skin}. The interaction radius is the maximum distance between two particles measured from centre to centre over which two particles can interact, i.e. particle length L in our system. The second radius, the skin radius, is the maximum distance between particle centres over which the particles are considered neighbours, see Figure 7.

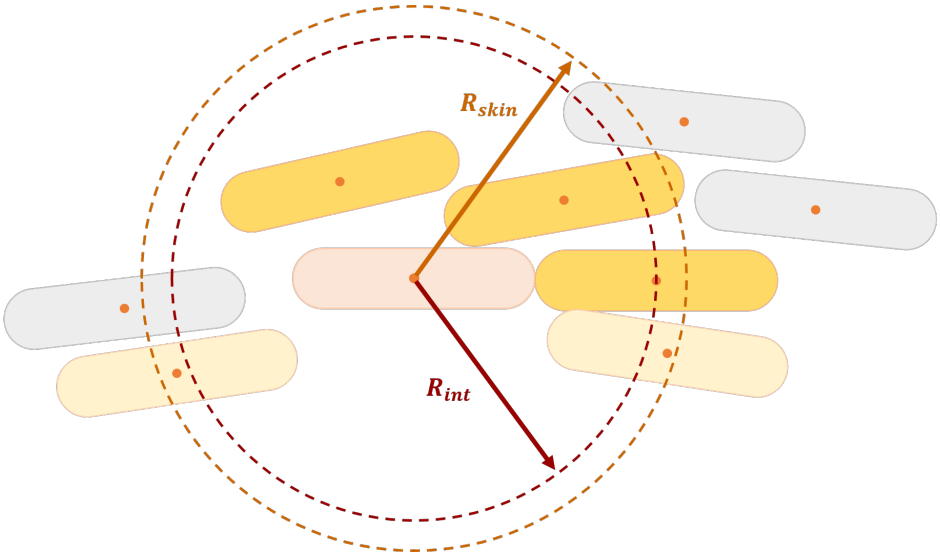


Figure 7: Particle with interaction radius R_{int} and skin radius R_{skin} illustrating which particles are included in the Verlet list (in yellow) and which are not (in grey), based on the centre positions. Here, we make a distinction between particles within the interaction radius (in dark yellow) and particles that cannot interact with the centre particle (in light yellow).

A necessary requirement is that the Verlet list contains at least the potentially interacting particles, $R_{\text{skin}} \geq R_{\text{int}}$, and that the Verlet list contains each particle at most once, $R_{\text{skin}} < S$, where spatial parameter S is half the width of the simulation environment.

To determine when to update the Verlet list, we look at the two largest displacements Δx_i and Δx_j the particles made in a single time step. We sum these two for every time step, until the cumulative sum over time gets larger than 95% of the difference between skin and interaction radius, $\sum_t (\Delta x_{i,t} + \Delta x_{j,t}) \geq 0.95(R_{\text{skin}} - R_{\text{int}})$, after which we update all Verlet lists. In this way, even when the two fastest particles move directly towards each other, they will not get in each other's region of interaction, assuming they do not exceed the 5% margin. We chose a skin radius $R_{\text{skin}} = 1.05R_{\text{int}}$, since this proved to be small enough to ensure only a small number of neighbours n, thereby reducing the complexity to $\mathcal{O}(nN)$, and large enough to limit update frequency, which comes at a computational cost with complexity $\mathcal{O}(N^2)$.

3.1.3 Stack vs heap memory

In programming, there is something called the ninety-ninety rule of optimisation, otherwise referred to as the more general Pareto principle, which states that 80% of the resources is consumed by 20% of the program.

Enter heap allocation. There are two ways of allocating memory in a computer: on the stack, which is small but fast memory, or on the heap, which is larger but slower. The custom vector class our predecessor built for computing linear algebraic operations stored its objects on the heap. We profiled the code using Very Sleepy [35], which showed that the allocation and deallocation of memory accounted for 90% of the computation time. Since the majority of the vector objects are used to store temporary intermediate results, they do not need to coexist in memory all at once, meaning that they do not require a lot of memory. Due to the small memory requirement, we can rewrite our vector class using the faster stack allocations, see Appendix A.3.

3.1.4 Miscellaneous

Lastly, there were some minor improvements like enabling level -O2 optimisation for the compiler, streamlining the code at compile time, reducing both computation time and compilation time.

Furthermore, the passing of variables by reference, i.e. only copying their address in memory instead of copying the whole object, was implemented more extensively. This reduces the memory needed, as well as the execution time, since memory allocation takes time.

The last performance improvement was regarding the I/O stream, specifically writing the simulation data to a file; a task that comes at the cost of computational overhead. Previously, the data was written to a file every time step, saving RAM, but in turn adding this overhead to the computation time. We solved this by storing the data in RAM and only writing it to a file after a specified number of time steps. For large simulations, containing in the order of 10^4 particles, the data was written to a new file for every 1000 saved time steps. This results in JSON files of around 2GB each, which can be handled easi by even an average computer, therefore justifying the computation speed vs memory trade-off.

3.2 Data analysis

To analyse our simulated system, we compute a few parameters that characterise the emergent collective dynamics. As previously stated, we perform all our data analysis using Python, and similar to the raw simulation data, we store all processed data in JSON formats.

Apart from analysing the simulated system, we also analyse the performance of the code.

3.2.1 Polar order parameters

We consider two parameters to quantify the polar order of the system; the local order parameter ϕ_{local} and the global order parameter ϕ_{global}. Each particle has a certain polarity, determined by its normalised swimming direction $\hat{\mathbf{u}}$. By calculating the inner product between the directions of the particles, we can define the local order parameter ϕ_{local} and global order parameter ϕ_{global} respectively as

$$\phi_{\text{local}} = \frac{1}{N}\sum_{i}^{N}\frac{1}{n_i}\sum_{j}^{n_i}\hat{\mathbf{u}}_\mathbf{i}\cdot\hat{\mathbf{u}}_\mathbf{j}, \tag{45}$$

$$\phi_{\text{global}} = \frac{1}{N^2}\sum_{i,j}^{N}\hat{\mathbf{u}}_\mathbf{i}\cdot\hat{\mathbf{u}}_\mathbf{j}, \tag{46}$$

where n_i is the number of neighbours of particle i.

3.2.2 Nematic order parameter

An additional measure to quantify the order of our system is the nematic order parameter ϕ_{nematic}, adopted from liquid crystal theory. This quantity does not consider the inner product between the reciprocal particle directions. Instead, it is a measure for the alignment of the particles with a general director. We are interested in the nematic order in the tube environment due to the intrinsic direction of the cylindrical shape. For a tube with its longitudinal axis along the z-axis, the expression for the nematic order parameter becomes

$$\phi_{\text{nematic}} = \frac{1}{N}\sum_{i}^{N}\frac{3|\hat{\mathbf{u}}_i\cdot\hat{\mathbf{z}}|^2-1}{2}, \tag{47}$$

where $\hat{\mathbf{z}}$ is our general director; the unit vector along the longitudinal axis of the tube.

3.2.3 Mean-squared displacement

In statistical mechanics, the mean-squared displacement (MSD) is a commonly used quantity to study the dynamics of a system. The MSD is determined based on the displacement of a particle from a reference point at the time t_0:

$$\text{MSD}(t) = \frac{1}{N}\sum_{i}^{N}\|\mathbf{x}_\mathbf{i}(t)+\mathbf{\Delta}_\mathbf{i}(t)-\mathbf{x}_\mathbf{i}(t_0)\|^2. \tag{48}$$

We account for the periodicity of the system by adding the vector $\mathbf{\Delta_i}(t)$, which elements each are multiples of the periodic side-to-side distance, based on how many times the particle has crossed since the start time t_0.

It is only meaningful to compute the MSD from the time step t_{equi} when the system has reached a certain equilibrium, and the initial transient behaviour has passed. Therefore, we choose our reference point at time $t_0 = t_{\text{equi}}$. Due to the stochasticity of the system, t_{equi} is different for every simulation. We calculate t_{equi} based on the plateauing of the local order parameter, as described in Appendix A.1.

To get even more data from a single run, we consider the MSD of a single time interval at different time steps in the simulation. We can only do this if we take independent time intervals, for example for

$$\text{MSD}(t_2) = \frac{t_2 - t_0}{t_f}\Big(\|\mathbf{x}(t_2) - \mathbf{x_i}(t_0)\|^2 + \|\mathbf{x}(t_4) - \mathbf{x_i}(t_2)\|^2 + ... + \|\mathbf{x}(t_f) - \mathbf{x_i}(t_{f-2})\|^2\Big), \tag{49}$$

resulting in an average MSD over $\frac{t_f}{t_2 - t_0}$ intervals. As a result, the MSD for short times will be more accurate due to the abundance of time intervals, whereas the MSD for the second half of the simulation is less accurate due to the inability to average over multiple time intervals.

3.2.4 Reynolds number

The Reynolds number Re is a dimensionless measure in fluid dynamics used to quantify the degree of turbulence or laminarity of flow. low Reynolds numbers correspond to laminar flow and high Reynolds numbers correspond to turbulent flow.
Usually, when we talk about the Reynolds number, we talk about fluid flow. However, in our simulation, we do not have a flowing fluid. On the other hand, that merely depends on your perspective. From the perspective of an individual swimming particle, it is not him but the surrounding fluid that is moving. Therefore, we can consider the Reynolds number for an individual particle, where we take the speed of the particle as the speed of the surrounding flow from its perspective, resulting in the expression for the Reynolds number

$$\text{Re} = \frac{\rho\overline{\|\mathbf{v}\|}D_{E.coli}}{\eta}, \tag{50}$$

where ρ is the density of the simulated fluid environment[7], $\overline{\|\mathbf{v}\|}$ is the average speed of the particles, $D_{E.coli}$ is the diameter of an *E. coli* and η is the dynamic viscosity of the medium[7].

3.2.5 Polar defect detection

In addition to the previous quantities, we can also investigate the defect dynamics, adopted from the field of liquid crystal theory. In this thesis, we only give a theoretical description on the defect analysis. The results of this analysis for the 2D version of our system can be appreciated in Jonathan Faasse's Bachelor thesis [36].

Because our particles swim in a certain direction, and therefore have a certain polarity, we are interested in the polar defects. To detect these polar defects, we first need to detect the nematic defects, for which we use the rotational angle method of Zapotocky et al. [37].
In this method, we consider three particles and project their directions as lines onto a unit circle, such that centres coincide with the centre of the circle. We start at one of the ends of an arbitrary line l_1. The first step is to travel over the unit circle to the nearest end of another line l_2. Subsequently, we take the shortest path to the end of the last line l_3, and finally, we return to the nearest end of the initial line l_1, see Figure 8a2-d2. If we ended up at the other side of l_1, then the triplet forms a nematic defect, see Figure 8b2-d2.

[7]The simulation medium is water at 20 C, resulting in a density $\rho = 997\text{kg/m}^3$ and a dynamic viscosity $\eta = 1.0016 \cdot 10^{-3}\text{Pa s}$.

The next step is to determine whether the nematic defect is also a polar defect. For that, we proceed in a similar fashion, but take the particle's polarity into account. Again we project the directions of the triplet on the unit circle, but instead of lines, we will represent the directions as arrows. We start at the arrowhead of an arbitrary particle p_1. From here, we travel over the unit circle to the closest arrowhead, with the corresponding particle p_2. Subsequently, we take the shortest path to the arrowhead of the last particle p_3, and finally, we return to the arrowhead of the initial particle p_1, see Figure 8b3-d3. If we ended up making a full revolution around the unit circle, then the triplet forms a polar defect as well, see Figure 8c3,d3.

Lastly, we can determine the sign of the polar defect. For that, we go back to our triplet of particles. We start with the same particle p_1, and once more, we travel along the particles, in the same order as before. If this trajectory follows the same direction as the revolution with the projected arrows, then the sign of the polar defect is plus, see Figure 8d4. However, in the other case, where the revolutions are opposite, the polar defect gets a minus sign, as is the case in Figure 8c4.

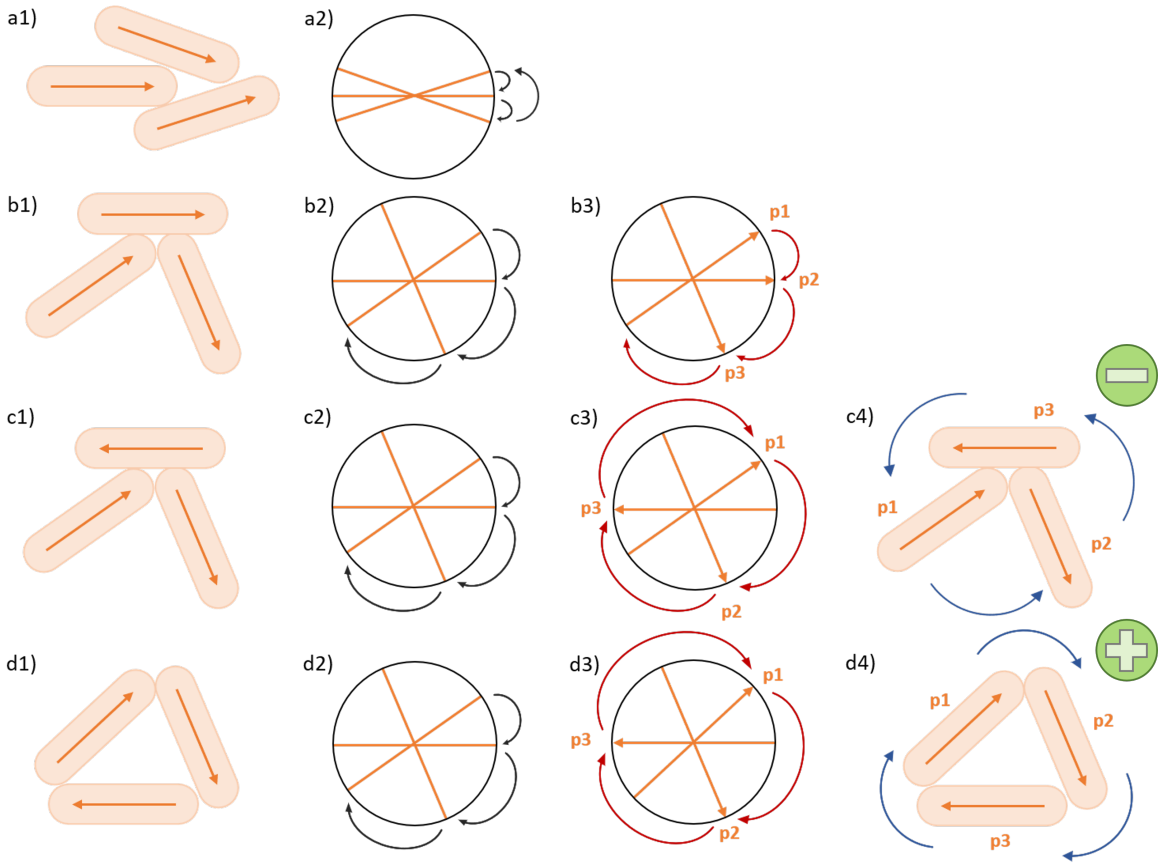


Figure 8: Polar defect detection in 4 steps (1-4) for four cases (a-d). Step 1, find a triplet of particles. Step 2, nematic defect determination: Projecting the particles' longitudinal axes on a circle & traversing the circle with the smallest steps. In case we end up on the opposite side, we have a nematic defect (b2-d2). Step 3, polar defect determination: Projecting the particle directions on the circle & traversing the circle with the smallest steps. In case we traverse the full circle, we have a polar defect (c3 & d3). Step4, sign determination: traversing the triplet. In case the traversing directions are the opposite/same, we have a minus (c4)/plus (d4) sign for the polar defect, respectively.

3.3 Sweep parameters

There are many independent parameters we can modify in the system we built. In an ideal world, we would like to do many simulations and vary all these parameters to study how each specific parameter affects the behaviour of the system. However, due to limited time, we need to limit our parameter space accordingly. Therefore, for our simulations, we focus only on the density ρ, due to the importance of mechanical interactions, especially in high-density systems [3, 10].

Furthermore, we want to investigate the newly introduced features of the system; the different geometries and the adhesive interaction. Specifically for the tubular geometry, we want to investigate how the tube radius R_{tube} affects the dynamics of the system.

The adhesive interaction is characterised by the adhesive spring constant k_{adh}. To reduce the parameter space, we set the particle-surface adhesive strength equal to the particle-particle adhesive strength $k_{\text{adh}} = k_{\text{adh,pp}} = k_{\text{adh,ps}}$, which we then will vary for different runs.

4 Results & Discussion

After optimising and expanding the model, we ran multiple simulations to marvel at the emergent behaviour. Similar to the previous sections, we will separate the results in two categories. First, we compare the computation time to see the effect of the optimisations. Furthermore, we look at how the systems scales with the number of particles N as well as for different densities ρ.

The second part of the results will be dedicated to the emergent dynamics of the system; in particular how the density and adhesive interaction affect the dynamics of the system. Subsequently, we also look at the effect of the size of the novel tubular geometry on the order in the system. All quantities are expressed in natural units, see section 2.5. The time t given in the plots is the time for a single saved step, earlier denoted as t_{save}, unless the units are specified.

4.1 Computation time

A substantial portion of the project was optimisation. To quantify our optimisations, we benchmarked our simulation of particles. To do a quick recap, the optimisations we implemented were the stack allocated vector class, enabling compiler optimisations, reducing the I/O-stream overhead by writing all the data to a file at once, and the Verlet list to reduce the complexity of $\mathcal{O}\big(N^2\big)$ to $\mathcal{O}(nN)$, where n is the length of the average Verlet list. We finish this quantification by comparing the performance of our current simulation in 2D with the previous simulating code, where we also take a look at the scaling of the simulation with the number of particles N.

Furthermore, we look at the effects of density on the computation time, and how the 3D periodic box and tube environment compare in performance. All simulations were run using 20 CPUs and for every saved time step, 300 intermediate calculation steps were performed. So a typical simulation with a number of saved time steps $Nt = 8001$, has $2.4 \cdot 10^6$ calculation steps.

4.1.1 Implemented optimisations

We implemented a new vector class, which stored its elements using faster stack memory instead of the slower heap memory. Figure 9 shows a decrease in computation time across all the vector class methods. Interestingly, this difference is ever greater when level -O2 optimisation is enabled, as shown in Figure 9b. Apparently, the compiler can optimise more rigorously using our new class. This could be due to the use of macros and the removal of the initialiser list.

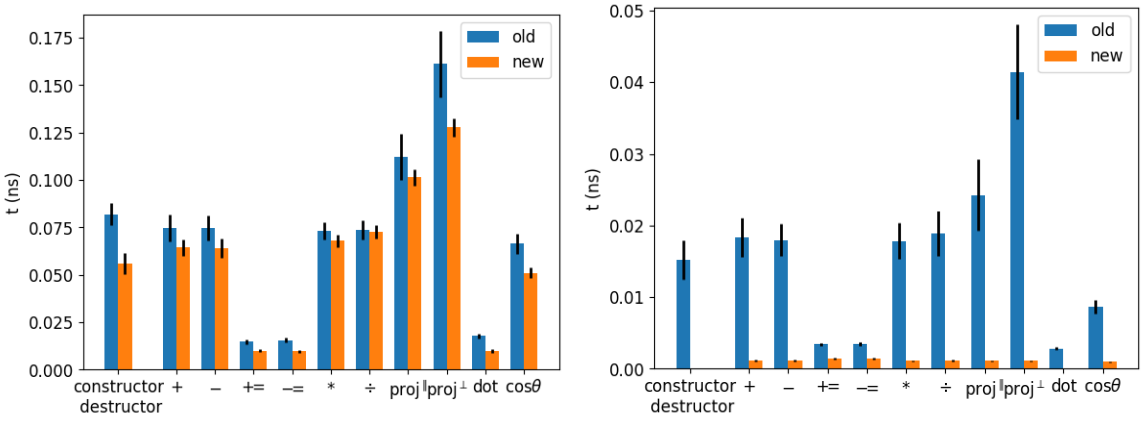
What stands out in Figure 9b is the zero computing time for the constructor and destructor, i.e. allocating and deallocating memory, as well as with the computation of the dot product. However, we believe this is due to the nature of our test. Since we do not use the results of these operations, the compiler dismisses this part of the code entirely, resulting in an average computation time of zero nanoseconds.

To further investigate the impact of the level -O2 optimisations by the compiler, we ran three simulations and plotted the average computation time in Figure 10. Here, we also show the impact of saving and writing the data every time step compared to only at the final time step t_f.

Enabling level -O2 optimisation by the compiler, the simulation performance increased approximately five-fold. A welcome but not entirely surprising result, given our previous tests of the vector class. However, it is nice to see these performance improvements translate to the simulation as well.

On the other hand, a peculiar phenomenon is the lack of difference in performance between writing the data every time step compared to saving the data at once. We would have expected the latter to take significantly less time to compute. One explanation could be that the data is first written to the local node on the cluster, instead of immediately to the hard drive. This would mean that both methods effectively act the same way, namely by only writing the data to the hard drive after completion of the program.

In the end, we chose a hybrid approach where the program writes the data to the hard drive in batches of a 1000 time steps. This way, we avoid the risk of a stack overflow where our program would demand more RAM than the system has to offer but still allows us to look at intermediate results.



(a) no compiler optimisation, level -O0

(b) maximal compiler optimisation, level -O2

Figure 9: The average computation time t for a single operation for all methods in the renewed vector class, compiled in both a) debug mode as well as b) release mode, with corresponding standard deviations. The time t is averaged over three loops of each 10^9 iterations.

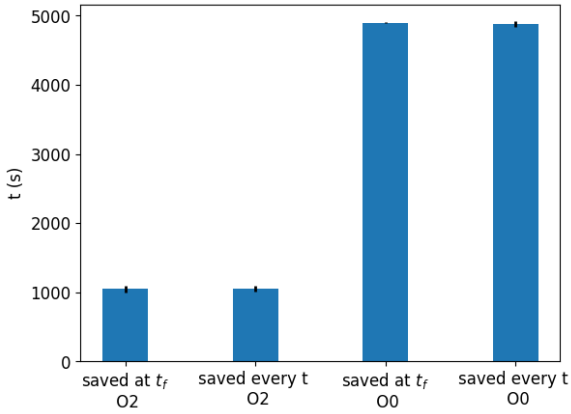


Figure 10: A comparison in computation time t in seconds between simulations with level -O2 compiler optimisations and without compiler optimisations, i.e. level -O0, where the particle data is saved either every time step or only at the final time step t_f. The shown values are averages with corresponding standard deviations of three 3D periodic box simulations, with $N = 10^3$, $Nt = 10^3$, $\rho = 0.5$.

4.1.2 Comparing simulation performance

In this section, we compare the performance of the old simulation to that of our current simulation. Due to some memory and incompatibility problems, we were not able to run the old simulation in 3D. Therefore, we only compare the different versions in 2D.

The result of our optimisations is shown in Figure 11. Here we plot the average computation time t in seconds for four 2D box simulations with different number of particles N, using the previous code (in blue) and the current code (in red). In addition, a third green line is shown displaying the CPU time for the current simulation. The CPU time is defined as the cumulative computation time of all working CPUs. Since we used 20 CPUs for this computation, the CPU times in green are roughly 20 times larger than their associated real times in blue.

To determine the scalability of our system, we do a linear fit through the computation times on a log-log scale, as shown in Figure 11b. Here, the slope of the fitted line tells us the scalability of the system. The first thing to note is the improved scalability of our current system with respect to the previous system. The linear fits through the data points give us a slope of 1.56 for the real time computation of the previous simulation, and a slope of 0.81 and 0.77 for the real time and CPU time of our current simulation.

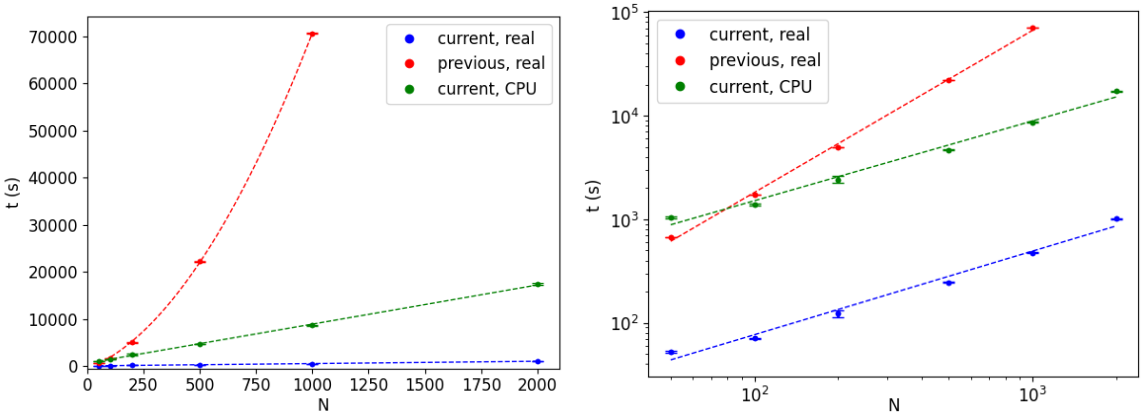


Figure 11: Comparison between the computation time t in seconds for different number of particles N, on a (a) linear and (b) log-log scale, including fitted polynomials.
In blue, we show the computation time of the current simulation with optimisations (linear fit: $t = 0.81N + 0.63$). In red, we show the previous simulation without any optimisations (linear fit: $t = 1.56N + 0.30$). In green, we show the cumulative computation time of all CPUs, for the current simulation, associated with the blue curve (linear fit: $t = 0.77N + 3.77$).
The computation times are averages over four 2D box simulations, with corresponding standard deviations. Each simulation had 3001 saved time steps Nt, corresponding with $9 \cdot 10^5$ calculated steps, a density $\rho = 0.5$, and was computed using 20 CPUs.

The fact that our current simulation has a lower slope and therefore scales better with N is not that surprising; we expected a difference in scalability due to the implementation of the Verlet list method. However, the resulting slopes are unexpected. Without the Verlet list method, the code should have a complexity of $\mathcal{O}(N^2)$, translating in a slope of 2 in the log-log plot. However, the resulting slope is lower. What is even more curious is the fact that the slopes of our real time and CPU time graphs are below 1, implying that our systems scale better than linearly. A linear scaling, i.e. a system with a complexity of $\mathcal{O}(N)$, would be a best-case scenario for our system. The Verlet list would never need to be updated; an operation with complexity $\mathcal{O}(N^2)$ and the total number of particles N would be much larger than the length of the Verlet list n. We know, however, that even though our Verlet list length is relatively minor compared to our number of particles, our system does require the Verlet lists to be updated. Therefore, we would expect the slopes to be between 1 and 2.

After some playing around with the profiler Very Sleepy CS and digging through the code, we found a possible explanation for the oddly optimistic scaling. The main performance bottleneck in the system is currently the creation of ghosts near the periodic boundaries. We profiled the system for the number of particles, which scales with the area of the 2D periodic box. However, the length of the boundaries scales with the square of the area. Hence, creating a dominant contribution of $\mathcal{O}(N^{1/2})$ to the scaling of the system and thereby decreasing the slope below 1. For the 3D system, the contribution of the periodic boundaries to the scaling of the system would be $\mathcal{O}(N^{2/3})$.

The profiler hinted towards the heap allocation and deallocation of memory every time a ghost is created. The ghost particle, based on the Particle structure, should be allocated on the stack, and the vector in which the ghosts are put should inherit the allocation type of its elements. Therefore, no heap allocation associated with ghost particles should occur.

We did, however, find a Particle object passed by value in a function concerning the boundary, creating a copy of the object instead of using the address in memory. This would partially account for the above described sub-one scaling of the program, however, there are still ghosts arrays created and deleted. Therefore, the periodic boundaries are still an issue to tackle. Correcting the passing by value reduced the number of heap allocations and sped up our calculations over two times in a small test case. This improvement would be less severe for larger simulations due to the relatively smaller box area. In the results shown here, this flaw is not corrected but our code should already be significantly faster in future use.

The main point to take away from this is the importance of passing by reference, instead of copy-

ing values. It is, however, still important for future researchers to look into this heap allocation of the Particle structure if they want to improve the performance of the simulation.

Some more context that should be given with Figure 11b is the fact that the computation with the previous code was not run in parallel, which it allegedly should be able to do. However, though provided with 20 CPUs, the real time and CPU were similar, indicating that it ran on only a single core. An attempt to parallelise this code appeared futile, and debugging would not yield anything since the current code already does run in parallel. However, to make an equal comparison, assuming the previous code was indeed correctly parallelised, we can compare the real computation time of the previous simulation with the CPU time of our current simulation. In this case, we still see the improved scalability, however, for $N = 50$, the previous simulation does run faster. This is probably because the Verlet list method only results in a computational advantage for large N. Else, it only contributes additional computation time due to the creation of the neighbour lists.

Lastly, we used a 2D box simulation for the performance comparison since the previous simulation code still had some bugs in its 3D simulation. However, in theory the difference in performance should be even more noticeable due to faster vector class, which is allocated on the stack instead of using the slower heap memory.

We can compare our simulation performance with other simulation software as well. Naylor et al. built a software package called Simbiotics [38] able to simulate up to 160 000 particles for 240 000 calculated steps in 38 hours, which is significantly faster than the 10 000 particles we simulate for 240 000 calculated steps in 15 to 65 hours depending on the density and geometry.

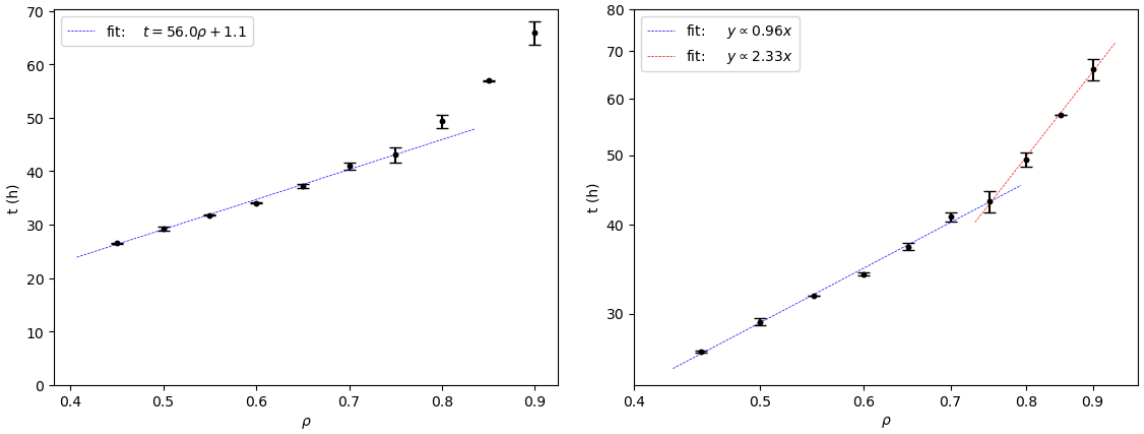
It should be noted that this comparison is hard to make. The Simbiotics simulation used a quarter of the processing cores but eight times the RAM. Furthermore, their system did not contain self-propelling particles, which require updating the Verlet list, nor was there a need for periodic boundaries, the main bottleneck of our system. Other differences are the additional multi-threading, better streamlined computation loops for more efficient parallelisation. Other authors report to simulate up to 40 000 particles [39], however, no computation time is given. Furthermore, comparing the performance with bacterial growth models would be arbitrary since these have a varying number of particles, slowing down as more particles are brought into existence.

4.1.3 Density and geometry

The density of the system and the geometry of the environment also influence the computation time of the simulation. Let us start by considering the density ρ, defined as the fraction of the cumulative particle volume over the volume of the environment. In Figure 12, we plotted the computation time for different densities in a 3D periodic box simulation, showing computation times ranging from 25 to 65 hours for a typical 10 000 particle simulation with 8001 time steps. For low densities the computation time seems to increase linearly, which is confirmed by the linear fit with a slope of 0.96 (in blue) in the log-log plot in Figure 12b for $0.4 < \rho \leq 0.75$. However, this linearity breaks for $\rho \geq 0.75$, where the slope of the fit function becomes equal to 2.33.

The increase of computation time with density is expected due to the increase of computed interactions with neighbouring particles. However, this does not explain the sudden increase in scaling of the system at $\rho = 0.75$. One could argue that the phase transition in the system (see section 4.2.1) affects the computation time. However, in that case, one would see a change in computation time per saved timed step throughout a simulation, which is not the case.

The only plausible explanation we can suggest has to do with the computational burden of the periodic boundary interactions with the heap-allocated ghost particles. In order to increase ρ while keeping N constant, we decrease the size of the periodic box accordingly. This results in box consisting for a relatively larger portion out of boundaries, for higher ρ. It could be, that the computational burden of the boundaries only becomes dominant for higher ρ, where the particle-particle interactions would be dominant for lower ρ.

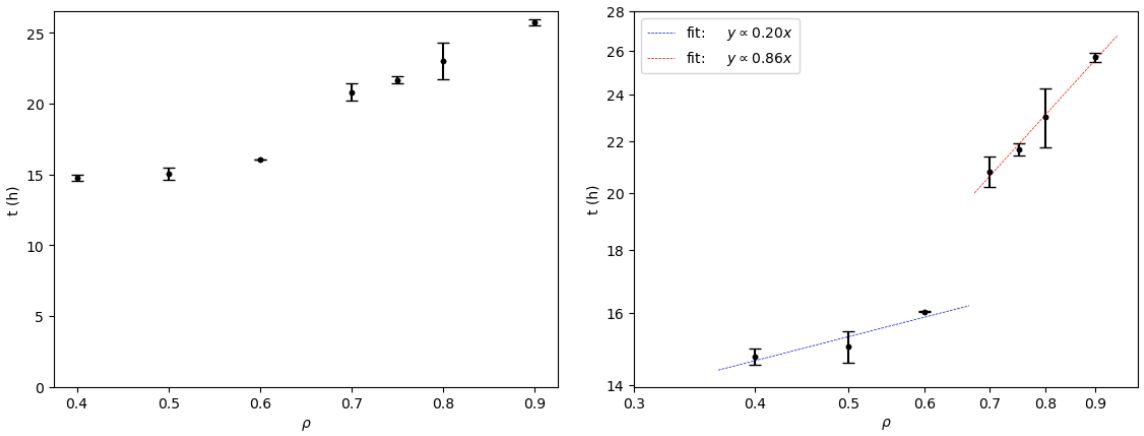


(a) linear scale

(b) log-log scale

Figure 12: The linear and non-linear regime of the computation time t for different densities ρ, for the 3D periodic box environment, with linear least-square fits. The computation times are averages over three simulations, with corresponding standard deviations. $N = 10^4$, $Nt = 8001$.

In Figure 13, we again plot the computation time t for different densities ρ, this time for the tube environment. Again we see a critical point, this time around $\rho = 0.7$ for which the computation time suddenly increases drastically.



(a) linear scale

(b) log-log scale

Figure 13: Computation time t for different densities ρ for a tube environment. The computation times are average over two simulations, with corresponding standard deviations. $N = 10^4$, $Nt = 8001$.

Moreover, it should be noted that the computation time is significantly lower for the tube environment, with computation times ranging from 15 to 26 hours for a typical 10 000 particle simulation with 8001 time steps. This is a performance improvement of a factor 1.5 to 2.5 compared 3D periodic box, depending on the density. This fits nicely in our earlier hypothesis of the boundary performance bottleneck. Only the two opposing sides of the tube are periodic while the curved area is not, making the limited effect therefore less apparent.

4.1.4 Tube radius

Lastly, we plotted the computation time of the tube geometry for different tube radii R_{tube} in Figure 14. However, we do not see a significant difference in computation time here.

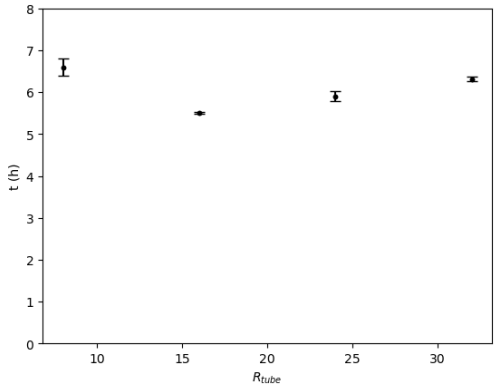


Figure 14: Computation times for the tube environment for different tube radii. $Nt = 3001$, $N = 10^4$ and $\rho = 0.5$

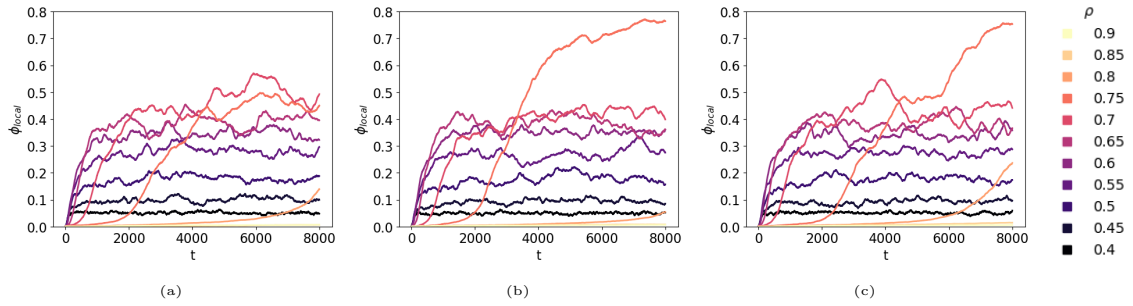
4.2 Local order parameter

As mentioned in section 3.2.1, the polar local order parameter measures the degree of alignment of an individual particle with its neighbours. We computed the local order parameter for both the 3D periodic box and the tube geometry.

4.2.1 3D periodic box

The local order parameter ϕ_{local} is plotted for different densities ρ for a 3D periodic box, see Figure 15. Here we see that in the range $0.4 \leq \rho \leq 0.75$, higher densities correlate to a higher local order parameter, which is in line with previous results from 2D simulations [23].

Let us take a closer look at the curve associated with $\rho = 0.75$. In Figure 15b and 15c in particular, we see this curve standing out, with a significantly higher local order at the final time step. Furthermore, we observe a delay period before ϕ_{local} drastically increases and equilibrates. A willing spirit could even recognise an S-curve.

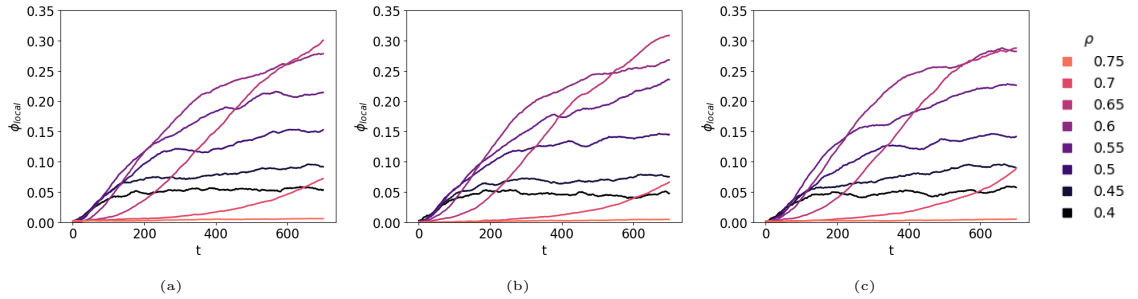


(a) (b) (c)

Figure 15: Local order parameter ϕ_{local} as a function of time steps t, plotted for different densities ρ, for three different 3D periodic box simulations. $N = 10^4$

With this behaviour in mind, we can look at the curves with similar densities, $\rho = 0.6$ and $\rho = 0.8$. For $\rho = 0.6$ we see a similar smaller period of delay before ϕ_{local} increases. On the other hand, the curve $\rho = 0.8$ stays flat for the largest part of the simulation but increases slightly at the end. This suggests that the period of delay is larger for larger densities while the local order parameter might surpass those of lower densities.

To back this theory up, we zoom in an order of magnitude to the range $0 \leq t \leq 700$ in Figure 16. Here, we can see similar delays for lower densities like $\rho = 0.6$ and $\rho = 0.65$, where the shape of these curves even suggests an S-curve in Figure 16c.



(a) (b) (c)

Figure 16: A zoomed in version of 15, with the local order parameter ϕ_{local} in the time step range $t < 700$, plotted for different densities ρ, for three different 3D periodic box simulations. $N = 10^4$

To further investigate the relation between the density ρ and the local order parameter ϕ_{local}, we calculated the average local order parameter with standard deviation for all three 3D box runs over the interval $[t_{\text{equi}}, t_f]$ and plotted the results in Figure 17. Here, the seeming relation between the density of the system and the local polar order becomes more apparent for a system where the local order has equilibrated ($\rho \leq 0.75$). When we do a linear fit through the data, we are presented with the equation $\phi_{\text{local}} = (1.49 \pm 0.09)\rho - 0.56$, with $R = 0.99$, which would predict a system with a maximum density $\rho = 1.0$ to closely approach the maximum local order $\phi_{\text{local}} = 1.0$, under the assumption the system would be able to equilibrate. However, in practice, such a system could probably only equilibrate after an infinite amount of time if it would not be jammed entirely; a scenario we could confirm by computing a Reynolds number equal to zero. However, this extreme case still poses a pleasant prospect to the validity of this relation. A maximal density would imply optimal packing of the particles, which would be in an orderly aligned structure, with maximal non-polar order and possibly with maximal polar order, after equilibrating.

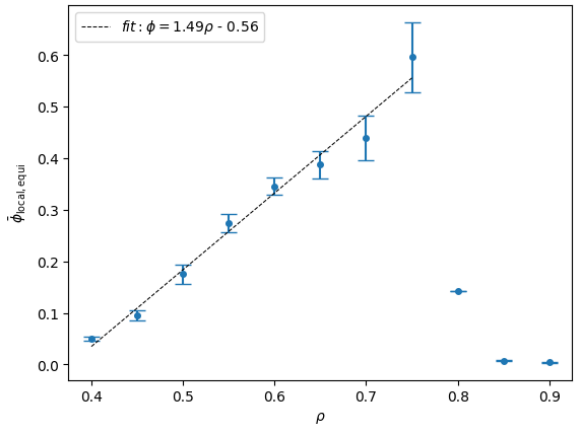
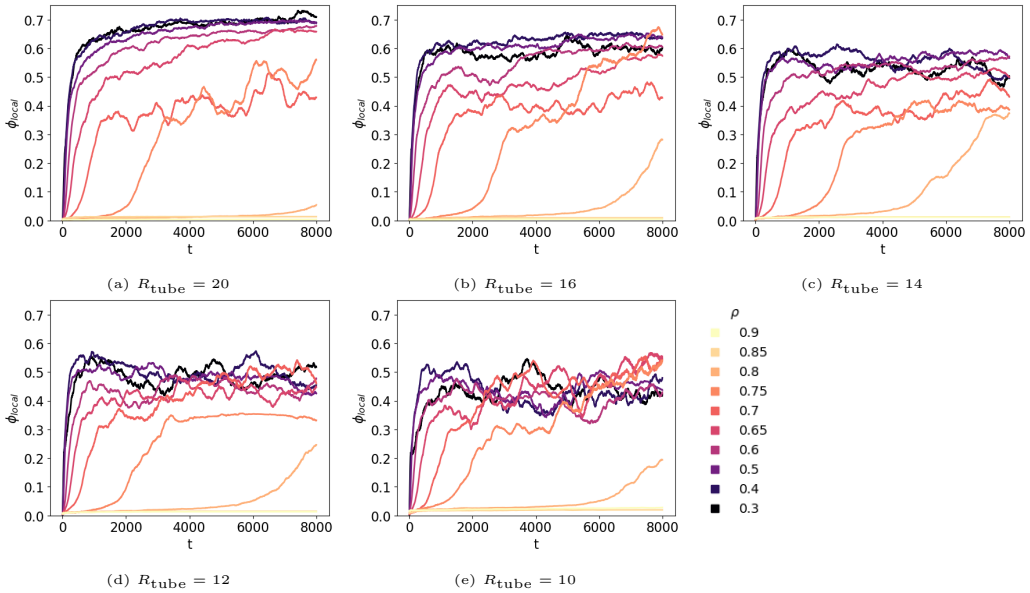


Figure 17: The average local order parameter $\bar{\phi}_{\text{local}}$ over interval $[t_{\text{equi}}, t_f]$ plotted against the density ρ with the standard deviation, averaged over three runs, in the 3D periodic box geometry, with $N = 10^4$. The points for which the system has reached equilibrium are included in a linear fit.

4.2.2 Tube

In addition to the 3D periodic box, we ran similar simulations in the tube environment. On the one hand, to assess if the density similarly affected the local order parameter, on the other to investigate if the tube radius R_{tube} had a significant effect on this quantity.

Let us start by evaluating the effect of the density in the tube environment. Similar to in the 3D periodic box environment, we see that systems with a lower density reach their equilibrium faster. For high densities like 0.75, there is a delay period before the local order increases, as shown in Figure 18. Moreover, we again see that for $\rho = 0.8$, the local order parameter slightly increased but the moment of increase is again stochastic.



(a) $R_{\text{tube}} = 20$ (b) $R_{\text{tube}} = 16$ (c) $R_{\text{tube}} = 14$

(d) $R_{\text{tube}} = 12$ (e) $R_{\text{tube}} = 10$

Figure 18: Local order parameter ϕ_{local} against saved time steps t, plotted for different tube radii R_{tube}, where the different colours indicate different densities ρ. $4009 \leq N \leq 5725$

On the other hand, we also see a significant difference with the 3D periodic box. Namely, the local order is much higher for densities $\rho \leq 0.7$. This fact becomes especially apparent when we look at Figure 19, showing the average local order at equilibrium $\phi_{\text{local,equi}}$, i.e. over the interval $[t_{\text{equi}}, t_f]$, for the different densities and tube radii. The data points for $\rho = 0.75$ are hard to interpret due to the two-step phase transition the system undergoes, making it harder to determine the moment of equilibrium. In order to make a significant statement, the system would need to run longer, with an improved way to determine the moment of equilibrium, as discussed in section 5.2.1. Again, the high densities $\rho \geq 0.8$ show a non-equilibrated system and can therefore be ignored.

Furthermore, we see a relation where larger tube radii lead to a higher local order. A possible explanation could be that small tube diameters, the smallest being only 2.5 particles lengths, have too little space for the particle to manoeuvre and align in clusters. This would also explain the larger standard deviations due to more stochastic behaviour in narrower tubes, as can be observed when comparing the smoothness of the plots in Figure 18e and Figure 18a.

This theory would imply there exists some optimal tube radius for which the local order would be maximal since a system with an infinitely large tube radius would be the same as the periodic box. However, future research should confirm this by running systems with higher tube radii $R_{\text{tube}} \geq 20$.

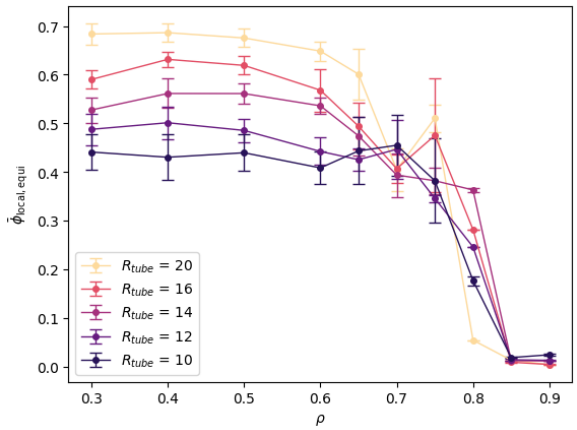
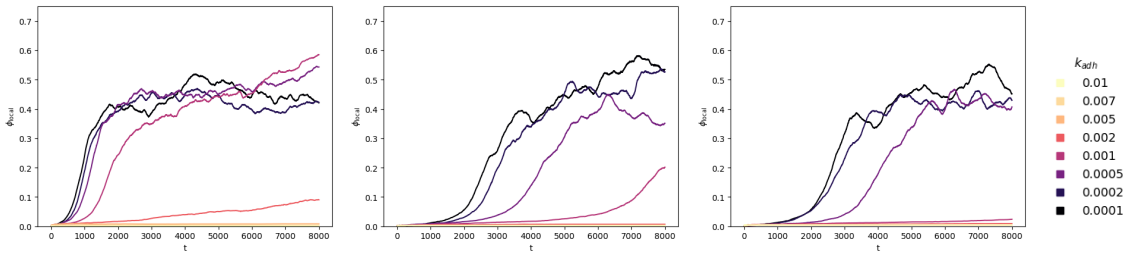


Figure 19: The equilibrated local order parameter $\phi_{\text{local,equi}}$ for different densities ρ and different tube radii R_{tube}.

4.2.3 Adhesion spring constant

The adhesion was implemented in the model. However, this introduced a new free parameter, the adhesion spring constant k_{adh}. We ran the system for different values of k_{adh} to study its effect on the dynamics of the system, see Figure 20. Initially, we did not have a good grasp of a plausible order of magnitude for k_{adh}, also because we did not find any guiding literature on particle-particle adhesion. Therefore, we chose for logarithmically distributed values between 10^{-4} and 10^{-2} times the amount of the repelling overlap spring constant k_{ov}.

As expected, the systems with a high k_{adh} have a low order due to the combination of random initialisation of particle directions and the adhesive effects resulting in stacks of particles with an even distribution of aligned and anti-aligned particle directions. Furthermore, the particles have difficulty moving due to the strong adhesion forces, resulting in the inhibition of a phase transition. Depending on the density, $k_{\text{adh}} = 0.002$ for $\rho = 0.7$ and $k_{\text{adh}} = 0.001$ for $\rho = 0.75$, are critical values where a change in order can occur, as seen in Figures 20a and 20b respectively.



(a) $\rho = 0.7$ (b) $\rho = 0.75$ (c) $\rho = 0.8$

Figure 20: Local order parameter ϕ_{local} as a function of time steps t, plotted for different densities ρ and different adhesion spring constants k_{adh}. $R_{\text{tube}} = 23, N = 10^4$.

Figure 21 summaries the time-dependent data into a single figure, considering the average local order at equilibrium. Here, the change in order around $k_{\text{adh}} = 0.001$ can be seen even more clearly. It should be noted, however, that data points at $\bar{\phi}_{\text{local,equi}} = 0.1$ and $\bar{\phi}_{\text{local,equi}} = 0.2$, are not of equilibrated systems, and therefore should be ignored in this figure.

It could be interesting to run longer simulations and more simulations, for systems with a k_{adh} around 0.001, to see where the shift in equilibration resides, as well as to investigate the existence of a relation between the adhesion spring and the order. More simulations with different k_{adh} would mean a higher resolution and would enable us to distil this potential relation, for which we currently have simply too few data points.

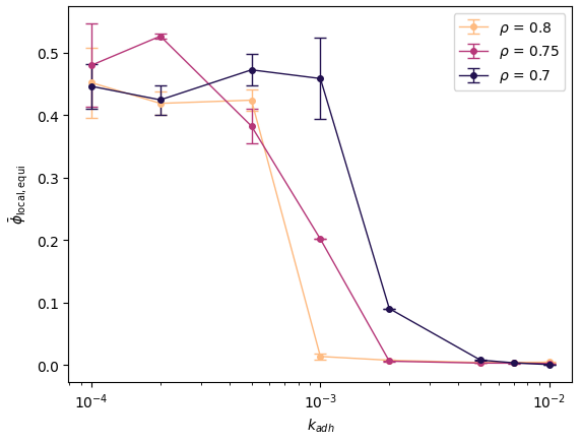


Figure 21: The equilibrated local order parameter $\phi_{\text{local,equi}}$ for different adhesion spring constants k_{adh} and different densities ρ.

4.3 Global polar order parameter

The global polar order ϕ_{global} is a measure for the reciprocal alignment between all particles while taking into account their polarity. We computed this quantity for both the 3D periodic box and the tube environment, plotted in Figure 22 and 23, respectively. Here, we see that the global order is notably smaller, compared to the local order parameter. This is expected since the system is initialised without a net direction while forming smaller flocks of particles, which increases the local order but not necessarily to the global order. Furthermore, we see some fluctuations for $0.75 \leq \rho \leq 0.85$ for the 3D periodic box environment and for $\rho = 0.7$ for the tube environment. It is not immediately clear what causes these fluctuations. However, it is notable that these fluctuations occur around the same densities where we saw the critical behaviour in the local order parameter. Apart from the fluctuation heights, we do not observe any notable density-dependent behaviour. We saw similar behaviour for the adhesion spring constant, with a few fluctuations, only for low adhesion spring constants but no notable correlations.

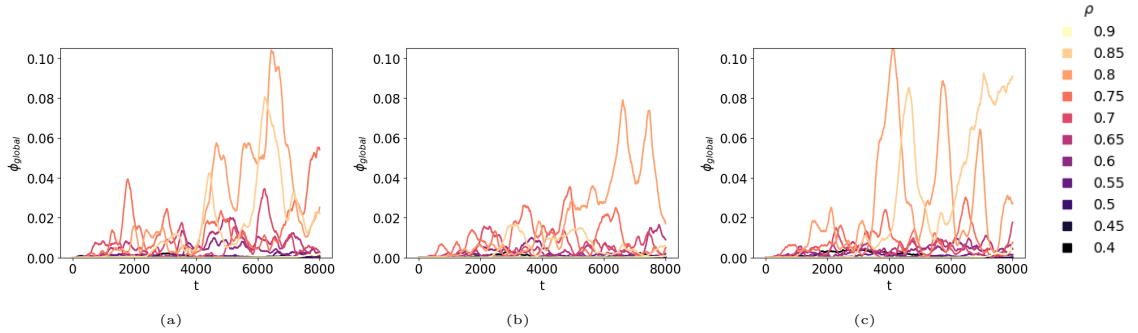


Figure 22: Polar global order parameter ϕ_{global} as a function of time steps t, plotted for different densities, for three different 3D periodic box simulations. $N = 10^4$

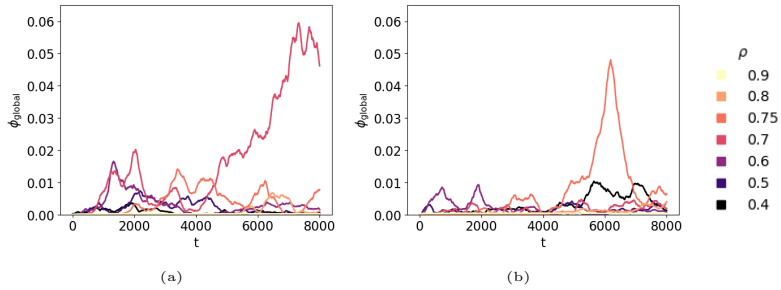


Figure 23: Polar global order parameter ϕ_{global} as a function of time steps t, plotted for different densities, for two different tube simulations. $N = 10^4$, $R_{\text{tube}} = 16$

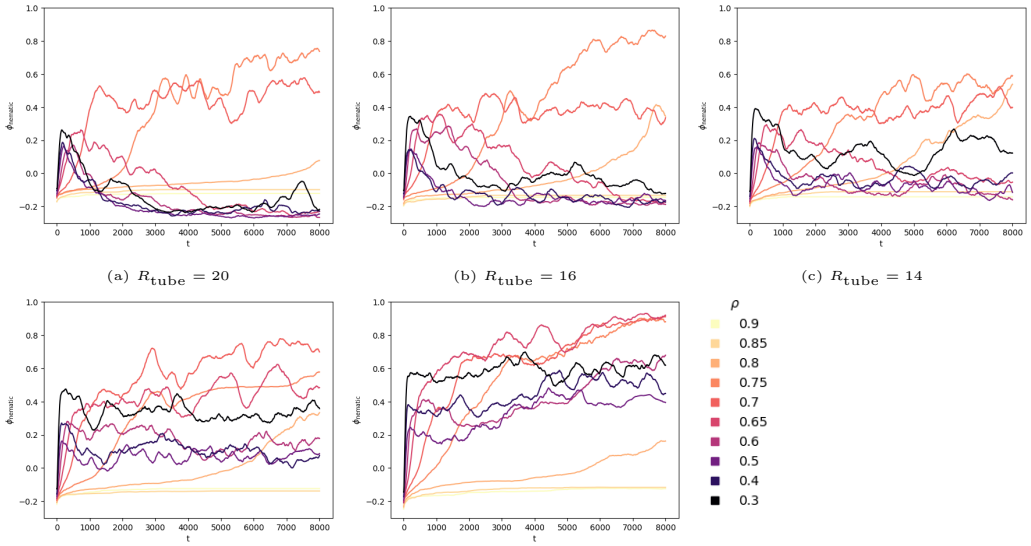
4.4 Nematic order parameter

The nematic order ϕ_{nematic} is a measure to quantify the alignment of the system with a given director. Since our 3D periodic box does not have an intrinsic direction, we therefore, only calculate ϕ_{nematic} for the tube environment, with the director along the longitudinal axis of the tube.

4.4.1 Density & Tube radius

In Figure 24, we see diverging behaviour for both different densities and different tube radii. Let us start by discussing the different densities. We see that for $\rho = 0.7$ and $\rho = 0.75$, the nematic order is relatively high for all tube radii. Though the contrast is most notable for a high tube radius, see Figure 24a.

This brings us to the lower densities. Here, the nematic order immediately increases at the start of the simulation but then dependent on the tube radius, either decreases again for higher radii (Figure 24a-24c) or increases slightly, which is the case for $R_{\text{tube}} = 10$, as seen in Figure 24e.



(a) $R_{\text{tube}} = 20$ (b) $R_{\text{tube}} = 16$ (c) $R_{\text{tube}} = 14$

(d) $R_{\text{tube}} = 12$ (e) $R_{\text{tube}} = 10$

Figure 24: Nematic order parameter ϕ_{local} as a function of time steps t, plotted for different tube radii R_{tube}, where the different colours indicate different densities ρ. $4009 \leq N \leq 5725$

In order to distil a relation, we plotted the average nematic order from the moment of equilibrium for different densities and tube radii, see Figure 25. Here, we can clearly see a relation where narrower tubes induce a higher nematic order for densities up until $\rho = 0.65$. One could argue that this relation exists for $\rho = 0.7$ as well. However due to the overlapping standard deviations, more simulations should be run to give a statistically significant value to this claim. This brings us to $\rho = 0.75$, where a relatively high and similar nematic order is observed. We can imagine that higher densities give rise to higher nematic orders simply because there is less space to move freely. This creates a statistic pressure for alignment due to the reciprocal collisions of particles, similar to the case for the previously discussed local order. In this time scale, $\rho = 0.75$ is the maximal density for which the system equilibrates. Higher densities may even lead to higher nematic orders over larger times scales. In contrast to the local order parameter, these higher orders could probably be reached over a shorter period due to the lack of polarity in the nematic order parameter.

An odd phenomenon is the seemingly minimal nematic order at $\rho = 0.5$. A possible explanation is that for low densities, the particles mainly reside at the walls of the tube. A particle against the wall will push itself parallel to that wall, where it has a maximal nematic order. Once against the wall, a particle cannot escape the wall if it does not collide with another particle due to the inability to turn on itself. Therefore, at low densities, where the environment is not crowded, particles are less likely to escape the wall, causing the system to have a higher nematic order. It should be emphasised that

this is merely a theory and not verified. Furthermore, it can only explain the higher nematic order at low densities, not why the ϕ_{nematic} is minimal at $\rho = 0.5$ in particular. However, it could be a middle ground between the hypothesised order inducing walls effect at low densities, which would diminish at high densities, and the previously described order inducing crowd effect, which would not be yet in effect at low densities.

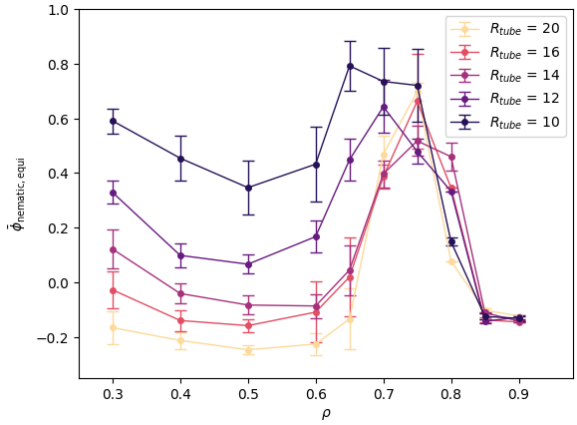
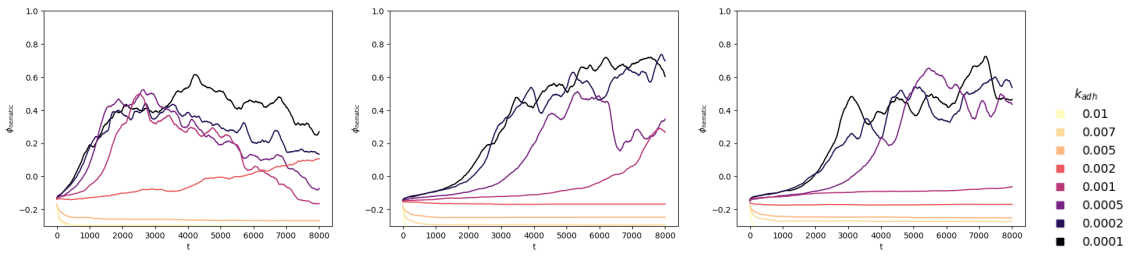


Figure 25: The equilibrated nematic order parameter $\phi_{\text{nematic,equi}}$ for different densities ρ and different tube radii R_{tube}.

4.4.2 Adhesion spring constant

When varying the adhesion spring constant k_{adh}, we find that for $k_{\text{adh}} \geq 0.001$, ϕ_{nematic} does not increase. This could be due to the chain formation of laterally stacked particles, where the chains align with the tube. Therefore aligning the particles perpendicular to the tube axis. However, this theory was not validated.

Figure 26a shows us something unexpected. For low $k_{\text{adh}} \leq 0.001$, the nematic order initially increases but gradually decreases over time. Further investigation is needed to find an explanation for this effect.



(a) $\rho = 0.7$ (b) $\rho = 0.75$ (c) $\rho = 0.8$

Figure 26: Nematic order parameter ϕ_{nematic} over time steps t, plotted for different densities ρ, where the different colours indicate different adhesion spring constants k_{adh}. $R_{\text{tube}} = 23$, $N = 10^4$

4.5 Mean-squared Displacement

In our journey to unravel the characterising dynamics of our system, we study the mean-squared displacement (MSD). Here, we redefine $t_0 = t_{\text{equi}}$, since we can only say something meaningful about the dynamics of an equilibrated system.

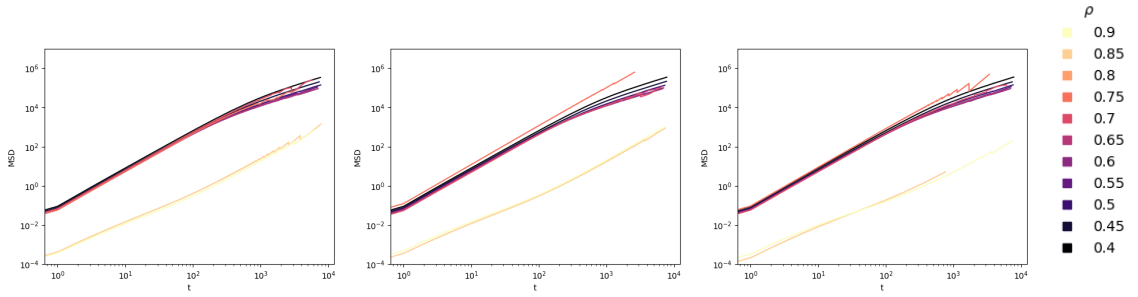
To determine the degree to which the particle behaviour is ballistic, we determine the slope m of a least-square linear fit for the MSD on a log-log scale. To make this slope reliable, we fitted for the interval $[t_0, t_{160}]$, for which we have averaged over at least 50 independent intervals.

4.5.1 Periodic box & tube

Here, we found a slope $\bar{m}_{\text{start}}$ of 1.9 for most systems that have reached equilibrium, i.e. $\rho \leq 0.75$, as shown in Table 3. This holds for both the periodic box, see Figure 27, as well as the tubular geometry, regardless of the tube radius, see Figure 28. This indicates that in the early stage of the system the motion is virtually ballistic: MSD $\propto t^2$. This is in line with what Marchetti et al. found for the motion of 2D self-propelling circular particles [40, 41].

Table 3: Slopes $\bar{m}_{\text{start}}$ and m_{tail}; the slope of the averaged MSD for interval $[t_0, t_{160}]$ and the slope of the non-averaged MSD for the last 1000 saved time steps. The slopes are averaged over three runs for the periodic box and over five runs for the tubular geometry, with corresponding standard deviations.

	Periodic box		**Tube**
ρ	$\bar{m}_{\text{start}}$	m_{tail}	$\bar{m}_{\text{start}}$
0.75	2.0 ± 0.0	2.0 ± 0.7	1.9 ± 0.1
0.7	1.9 ± 0.0	1.2 ± 0.8	1.9 ± 0.0
0.65	1.9 ± 0.0	0.9 ± 0.1	1.9 ± 0.0
0.6	1.9 ± 0.0	1.0 ± 0.0	1.9 ± 0.0
0.55	1.9 ± 0.0	1.2 ± 0.1	-
0.5	1.9 ± 0.0	1.0 ± 0.1	1.9 ± 0.0
0.45	1.9 ± 0.0	1.0 ± 0.0	-
0.4	1.9 ± 0.0	1.1 ± 0.0	1.9 ± 0.0



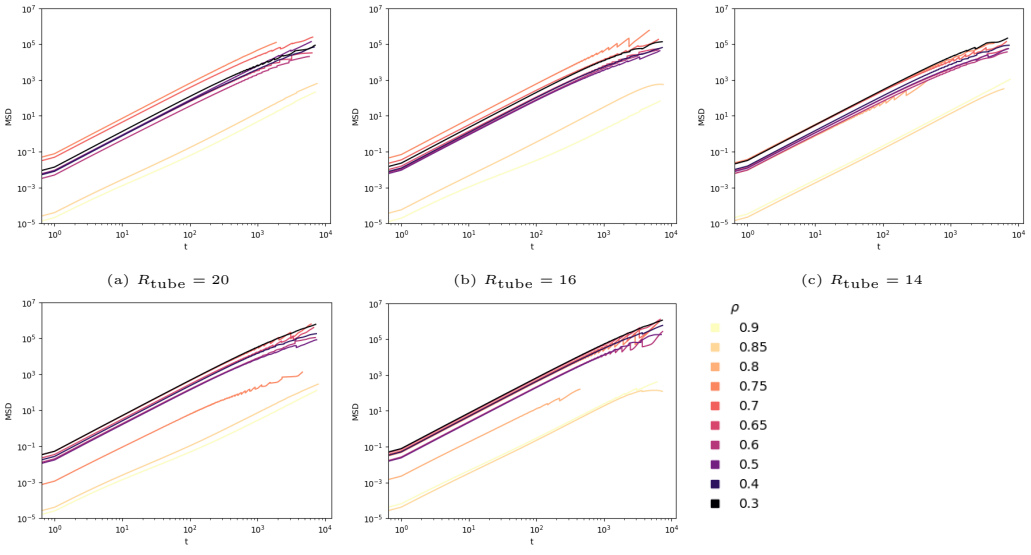
(a) (b) (c)

Figure 27: Mean squared displacement MSD measured from the point of equilibrium t_{equi} as a function of time steps t, for three different 3D periodic box simulations, where the different colours indicate different densities ρ. $N = 10^4$

Marchetti et al. found a slope of 1 at the end of their simulation. Despite not knowing if our system lives on a similar time scale, it still piqued our curiosity. Therefore, we investigate the slope m_{tail} at the tail of our MSD plot over the interval $[t_{7000}, t_{8000}]$. Due to the sawtooth pattern in the tail of our averaged MSD plot, more on that later, we consider only the normal MSD, not averaged over multiple time intervals, to investigate the slope at the final time steps. Bear in mind that the tail slope is therefore more prone to stochastic effects and, as a result, less reliable.

In the periodic box, we find that the slope at the end of our simulation decreases to values close to 1, as shown in Table 3, similar to the diffusive dynamics for passive Brownian motion, where MSD $\propto t$. This small slope is also consistent over multiple runs for lower densities, resulting in a small standard deviation. For $\rho = 0.7$ and $\rho = 0.75$ these results are less consistent. This could be due to the general critical behaviour of the system around this density. It very well could be the case that on this time scale, the system is still not entirely equilibrated, similar to the relatively high standard deviations in the local order parameter in Figure 17 for these densities.

A noticeable difference with Marchetti's study is that our system produces very similar MSDs for different densities. This is, however, given that the system has equilibrated, resetting the time as $t_0 = t_{\text{equi}}$. When considering the MSD for the full run, we see the system behaving differently for different densities, as shown in Figure 29.



(a) $R_{\text{tube}} = 20$ (b) $R_{\text{tube}} = 16$ (c) $R_{\text{tube}} = 14$

(d) $R_{\text{tube}} = 12$ (e) $R_{\text{tube}} = 10$

Figure 28: Mean squared displacement MSD measured from the point of equilibrium t_{equi} as a function of time steps t, plotted for different tube radii R_{tube}, where the different colours indicate different densities ρ. $4009 \leq N \leq 5725$

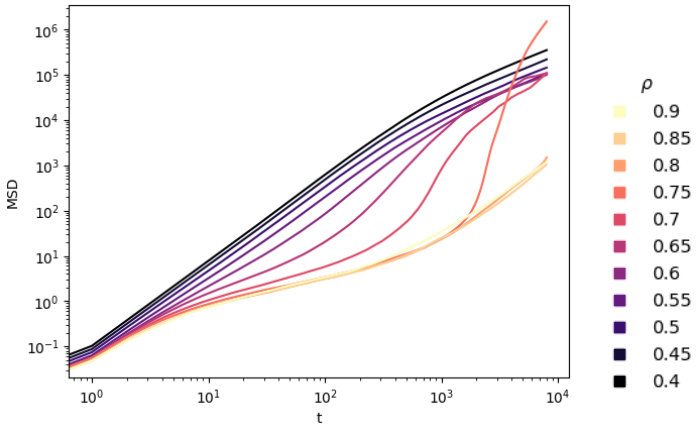


Figure 29: MSD over saved time steps t, for different densities ρ, for a periodic box.

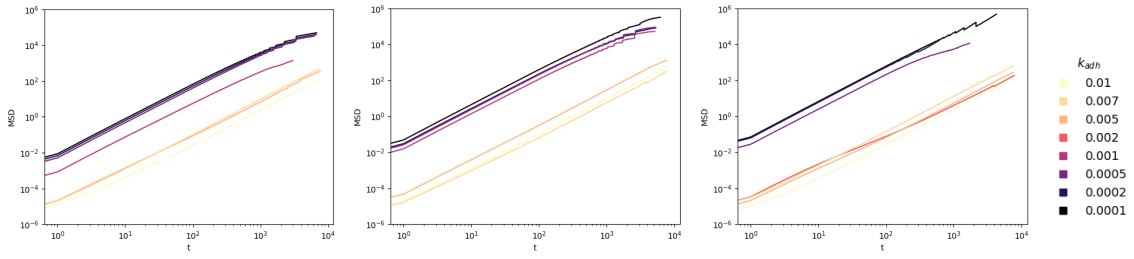
4.5.2 Adhesion spring constant

We also investigated the MSD for different adhesion spring constants $k$adh, as shown in Table 4. Here, we find similar results, with an overall slope of 1.9, indicating ballistic behaviour. Figure 30 shows a lower MSD for systems with $k_{\text{adh}} \geq 0.002$ that are either jammed or have not yet equilibrated.

We expect there to be a gradual change of MSD for different k_{adh}. We did not observe this because we only sampled k_{adh} in large logarithmically distributed steps. Only in Figure 30a do we see an MSD curve in the gap at $k_{\text{adh}} = 0.001$. We expect the k_{adh} around this value to produce more curves to fill the gap.

Table 4: Slopes $\bar{m}_{\text{start}}$ of the averaged MSD for interval $[t_0, t_{160}]$ for different adhesion spring constants k_{adh}. For some values we were not able to produce a fit, which are left blank here.

		$\bar{m}_{\text{start}}$		
k_{adh}	ρ	**0.6**	**0.7**	**0.75**
0.01		1.8	1.9	1.8
0.007		1.9	1.8	1.9
0.005		1.8	1.9	1.7
0.002		-	-	1.6
0.001		1.9	1.9	-
0.0005		1.9	1.9	1.9
0.0002		1.9	1.9	1.9
0.0001		1.9	1.9	2.0



(a) $\rho = 0.6$ (b) $\rho = 0.7$ (c) $\rho = 0.75$

Figure 30: Mean squared displacement MSD measured from the point of equilibrium t_{equi} as a function of time steps t, for three tube simulations, with different densities ρ, where the different colours indicate different adhesion spring constants k_{adh}.

4.5.3 Sawtooth pattern

The sawtooth patterns in the tail of the plots in Figures 27 to 30 can be explained by the averaging of the data points combined with the non-linear relation between the MSD and the time t. The particles increasingly cover more distance over time, resulting in the average MSD changing when the averaging interval is changed. This effect is most noticeable at the final interval changes. In particular, for the change in averaging interval $\Delta t = 4000$ to $\Delta t = 4001$, where in the case of the first interval there are two sample intervals, $[t_0, t_{4000}]$ and $[t_{4000}, t_{8000}]$, to average over in contrast to the second interval $[t_0, t_{4001}]$, of which there exists only one for a single run of 8001 saved time steps.

Since we determined the slope only based on time interval $[t_0, t_{160}]$, we do not encounter any significant effects of the sawtooth pattern in our slope determination.

In Appendix A.5, we present a comparison of the averaged and non-averaged MSD to illustrate the emergence of the sawtooth pattern.

4.6 Reynolds number

Our last quantity characterising the dynamics of our system is the Reynolds number. We know from literature that the typical Renolds number for an *E. coli* is in the order of magnitude 10^{-5} [29]. With this Reynolds number, we can confirm two things.

The first is qualitative in nature and concerns the self-consistency of our system. Specifically, if our assumptions on the drag force scaling linearly with velocity and neglecting the inertial forces in our equation of motion (eq. 18) are valid. This would only be the case of a low Reynolds number.

The second aspect is more straightforward and would implicitly validate our first point as well: our computed Reynolds number matching the 10^{-5}.

Unfortunately, we do not find a Reynolds number $\text{Re} \propto 10^{-5}$ but instead $\text{Re} \propto 10^{-3}$ with outliers even an order of magnitude higher, see Figures 31 and 32. On the other hand, not all hope is lost, since this Re is still small enough to validate our assumptions for the equation of motion.

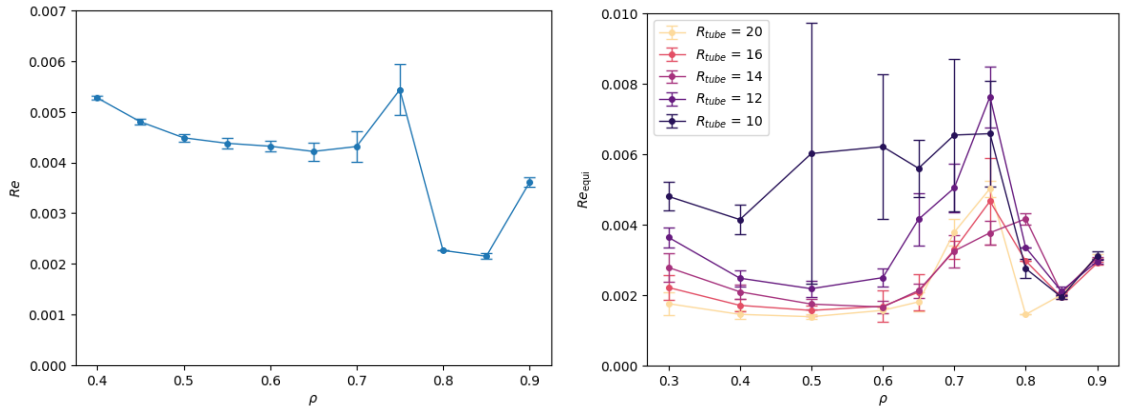
The question remains, why are we still a factor 100 removed from the actual order of magnitude? Let us consider the Reynolds eq. 50. The only varying parameter is the speed. The other parameters are all determined by the environment and the dimensions of the particle. The combination of the constant parameters approximately equals one. Therefore the order of magnitude of the Reynolds number is solely determined by the average speed $\overline{\|\mathbf{v}\|}$, which should comply with the typical swimming speed of *E. coli* $\|\mathbf{v}\|_{E.coli} = 25\mu m/s$ [42]. This means either the speeds in our simulation are incorrect or the conversion from natural to SI units is.

Let us first consider the conversion to SI units. We calculate the velocity in SI units as anyone who has ever followed a single physics course in their life: by dividing the covered distance, multiplied by conversion factor R_0, by the time of a single saved time step t_{saved} from eq. 44. However, in this t_0, other quantities are hidden, such as the viscosity η. In our calculations, we take the viscosity of plain water. However, bacteria secrete EPS, making the environment more viscous [43]. Moreover, the viscosity parameter η also reoccurs in the Reynolds expression. Therefore, an increase in η would result in a squared decrease for the Reynolds number. Unfortunately, we could not find the viscosity for an EPS-water mixture.

The other explanation for the higher Reynolds number is that the speeds in the simulation are already incorrect. This could be due to an incorrect self-propelling coefficient f_{sp}. We adopted the f_{sp} from previous work in the group [23]. Unfortunately, we could not find the basis of the specific value.

Let us subsequently discuss the relation between the density ρ, the tube radius R_{tube}, the adhesion spring constant k_{adh} and the Reynolds number. Figures 31a and 31b both show a small increase at low densities, as well as for higher densities around $\rho = 0.75$, where $\rho \geq 0.8$ can be ignored again since they did not equilibrate. For low densities, the increase of Re is to be expected because less collisions occur, therefore maximising the particle speed and therefore the Reynolds number. For high densities, the increase was unexpected because more collisions would occur. Perhaps not all collisions are created equal. Perhaps for high densities, the particles form larger aligned clusters, where the collisions are therefore mainly lateral. In contrast, the systems with a moderate density would form multiple smaller clusters with more opposing directions, where collisions constrain the speeds to a greater extent.

Figure 31b also shows a negative relation between the tube radius R_{tube} and the Reynolds number. This is probably due to the alignment inducing effect of the more confined space for lower R_{tube}, similar to the hypothesised effect for the nematic order in Figure 25.



(a) 3D periodic box environment

(b) Tube environment

Figure 31: The average Reynolds number Re over interval $[t_{\text{equi}}, t_f]$ plotted against the density ρ, in (a) the 3D periodic box environment and (b) in the tube environment for different tube radii R_{tube}.$N = 10^4$.

Figure 32 shows an almost linear negative relation between the adhesion spring constant k_{adh} and the Reynolds number Re, where the systems do not equilibrate for high k_{adh}. This negative relation is to be expected since the adhesion inhibits the free movement of the particles.

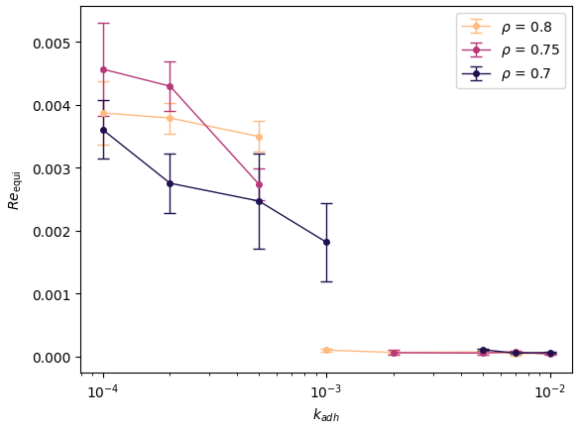


Figure 32: The average Reynolds number Re over interval $[t_{\mathrm{equi}}, t_f]$ plotted for different adhesion spring constants k_{adh} and different densities ρ, with $N = 10^4$.

5 Model discussion & Recommendations

In the previous section we discussed the dynamics of our bacterial colony in different geometries. Here, we will discuss the model itself and its implementation with the shortcomings and possible opportunities for improvement. This section is also intended as an initial guide for future modellers who wish to continue building on this project.

5.1 Model

Apart from adding new features, like the renewed drag force and the adhesive interaction, we also removed features: the particle's variable length and different ways of self-propulsion. In this section, we will discuss the reasoning behind our decisions and some adjustments to cater to a shortcoming we only discovered after getting our results.

5.1.1 Drag force

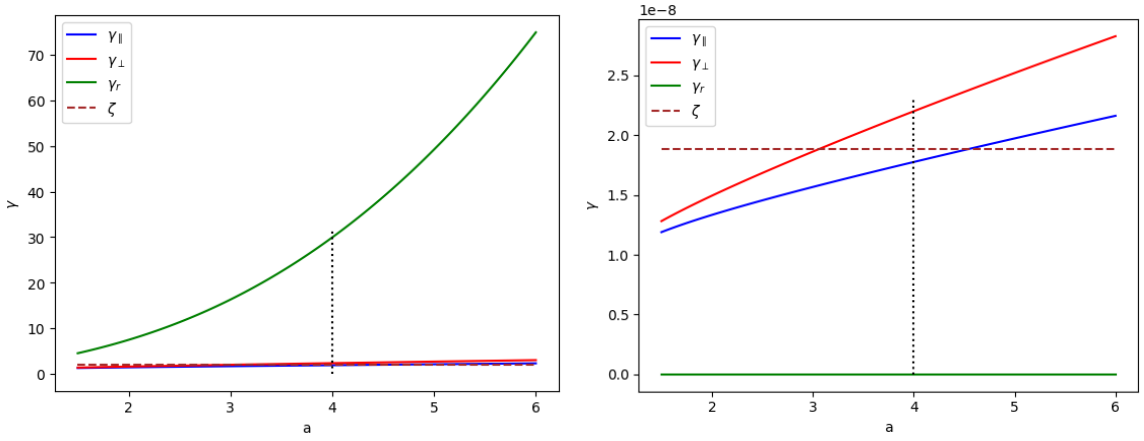
The main shortcoming in the previous model was the drag force, which was equivalent to that of two spheres, disregarding any length or orientational dependency. The current drag force is based on the experimentally derived drag coefficients of cylinders by Tirado et al. [26]. These coefficients are also validated by Bartuschat et al. [25]. In their findings, they state the translational drag coefficients for cylinders to be suitable for sphero-cylinders of the same length. The angular drag coefficients are less accurate, but still well within the same order of magnitude, resulting in only slightly slower angular velocities for sphero-cylinders of the same length.

Figure 33 shows the relation between the drag coefficients and the particle aspect ratio a, from eq. 21-23, where length L is substituted for the expression $2aR$ with radius R. As expected, all drag coefficients increase with the aspect ratio a.

The large difference between the rotational drag coefficient γ_r and the others can be explained by the difference in units which arises from the parameter L. The rotational coefficient has the cubic relation $\gamma_r \propto L^3$, whereas the other coefficients are linearly proportional to L. In natural units the cubic factor dominates the linear one, whilst in SI units the opposite is true due to the distance conversion factor $R_0 \propto 10^{-7}$. Despite the unit influence, it should be noted that this comparison of translational and rotational drag coefficients is not an entirely fair since we are comparing a relation of force with one of torque, where the latter contains an additional factor L, see eq. 26.

If we would compare the drag coefficients, in natural units, adjusting for the additional factor L due to torque, the rotational coefficient γ_r would still be dominant due to the additional factor L^2.

What came to a surprise to us, was how similar the previous drag coefficient ζ was to our current one. Particularly for our aspect ratio $a = 4$, the translational coefficients are remarkably similar. Furthermore, one might expect a lateral motion to be more difficult for a sphero-cylindrical shape than a forward one. However, due to the low Reynolds number the skin friction drag component, sometimes suitably called the viscous drag, is in the same order of magnitude compared to the pressure drag component. This fact also contributes to the similarity of the old drag coefficient of a sphere and the new drag coefficient.



(a) natural units

(b) SI units

Figure 33: A comparison of drag coefficients $\gamma_{\parallel}$, $\gamma_{\perp}$ and γ_r for the parallel, perpendicular and rotational drag, and the previous drag coefficient ζ, for different aspect ratios a, in both (a) natural units and (b) SI units. The dotted black line indicates the aspect ratio $a = 4$ used in our system.

5.1.2 Removal of the internal spring

Previously, our particle had an internal spring between its two spheres, enabling it to vary in length [23]. We removed this internal spring and in the following paragraph we will discuss our reasoning behind this decision. The problem with the internal spring was its incompatibility with our drag calculations and therefore with our computation of the displacement inspired by Bartuschat and Tirado [25, 26]. One can imagine a situation where one of the spheres would be stationary, while the other moves; a situation which is not compatible with our drag computation since our method assumes a rigid body. We prioritised the improved calculations for the displacement over the particle's ability to compress and therefore have removed the internal spring.

A consequence of this choice is that the self-propelling force F_{sp} cannot be unevenly distributed over the spheres anymore, removing the ability to distinctly model bacteria propelling by pushing and pulling. Luckily, previous research in the group already concluded this distinction to have a negligible effect on the particle dynamics [23].

5.1.3 Rotational motion

In our calculations on the displacement of the particle, we reduced the number of degrees of freedom (DOF) from three to two: the particle direction $\hat{u}$ and the orthogonal vector $\hat{v}$. The directions $\hat{u}$ and $\hat{v}$ are redefined at every time step by the net force on the particle according to eq. 27. Therefore, the reduction in DOF does not constrain its translational displacement. However, it does restrict the free orientation, i.e. rotational movement of the particle. For example, the particle cannot both move in a direction $\hat{v}$, orthogonal to its own direction $\hat{u}$, as well as rotate in the plane orthogonal to its movement. This makes the system non-physical. Therefore we need to introduce an additional direction $\hat{w} \equiv \hat{u} \times \hat{v}$, orthogonal to both $\hat{u}$ and $\hat{v}$. Taking this component into account, the expression for the angular speed ω becomes

$$\omega = \omega_1 + \omega_2 = \frac{L_{\text{inner}}}{2} \left\| \text{proj}_{\hat{v}} \left(\frac{\mathbf{F}_A - \mathbf{F}_B}{2} \right) \right\| + \frac{L_{\text{inner}}}{2} \left\| \text{proj}_{\hat{w}} \left(\frac{\mathbf{F}_A - \mathbf{F}_B}{2} \right) \right\|, \tag{51}$$

with an already existing component ω_1 describing the rotation in the uv-plane and a new component ω_2 describing the rotation in the uw-plane. This translates into the rotational displacement $\mathbf{\Delta x_r}$ and approximated rotational displacement $\mathbf{\Delta x'_r}$ accordingly:

$$\mathbf{\Delta x_r} = \frac{L_{\text{inner}}}{2} \Big(\Big[2 - \cos(\omega_1 \Delta t) - \cos(\omega_2 \Delta t) \Big] \hat{u} + \sin(\omega_1 \Delta t)\, \hat{v} + \sin(\omega_2 \Delta t)\, \hat{w} \Big), \tag{52}$$

$$\mathbf{\Delta x'_r} = \frac{L_{\text{inner}}}{2} \Delta t (\omega_1 \hat{v} + \omega_2 \hat{w}), \tag{53}$$

with a newly introduced component in the $\hat{w}$ direction.

There is no need to alter the expression for the translational displacement $\mathbf{\Delta x_T}$ as well. The translational displacement consisting of components $\hat{u}$ and $\hat{v}$ is already complete, since the direction $\hat{v}$ is defined by the direction of the net force $\mathbf{F_{net}}$

$$\begin{aligned}\mathbf{\Delta x_T} &\propto \text{proj}_{\hat{u}}(\mathbf{F_{net}}) + \text{proj}_{\hat{v}}(\mathbf{F_{net}}) \\ &\propto \text{proj}_{\hat{u}}(\mathbf{F_{net}}) + (\mathbf{F_{net}} - \text{proj}_{\hat{u}}(\mathbf{F_{net}})) \\ &\propto \mathbf{F_{net}}\end{aligned} \tag{54}$$

We do not expect large changes in the results, because the contribution of the $\hat{w}$ term is small. The displacement of the particle would mainly be in the particle direction $\hat{u}$, due to the self-propelling nature of the swimming particles. Therefore, the particle mainly does turning manoeuvres instead of spinning ones. The only small impact we can imagine is an increase in order, due to the reduced DOF.
This minimal impact is, however, under the assumption that torsional rotations around the particle axis $\hat{u}$ do not affect the dynamics of the system significantly; an implicit assumption we already made when designing the system.

5.1.4 Adhesive particles

To finalise our model discussion, we want to show a visual illustration of the newly implemented particle-particle adhesion. However, because it is hard to look inside a 3D colony, we present the states at $t = 1000$ for multiple colonies with different values for k_{adh} in the 2D system in Figure 34. In the figures associated with higher values for k_{adh} a flaw in the adhesion model becomes apparent; the particle-particle adhesion only applies to the lateral sides of the particles. This results in long strings of tightly side-by-side packed particles at high values for k_{adh}, as can be seen in Figures 34a to 34d. However, as a minimalist approach to simulate particle-particle adhesion, we still think this adhesive spring model is valid, though high spring constants should be avoided.

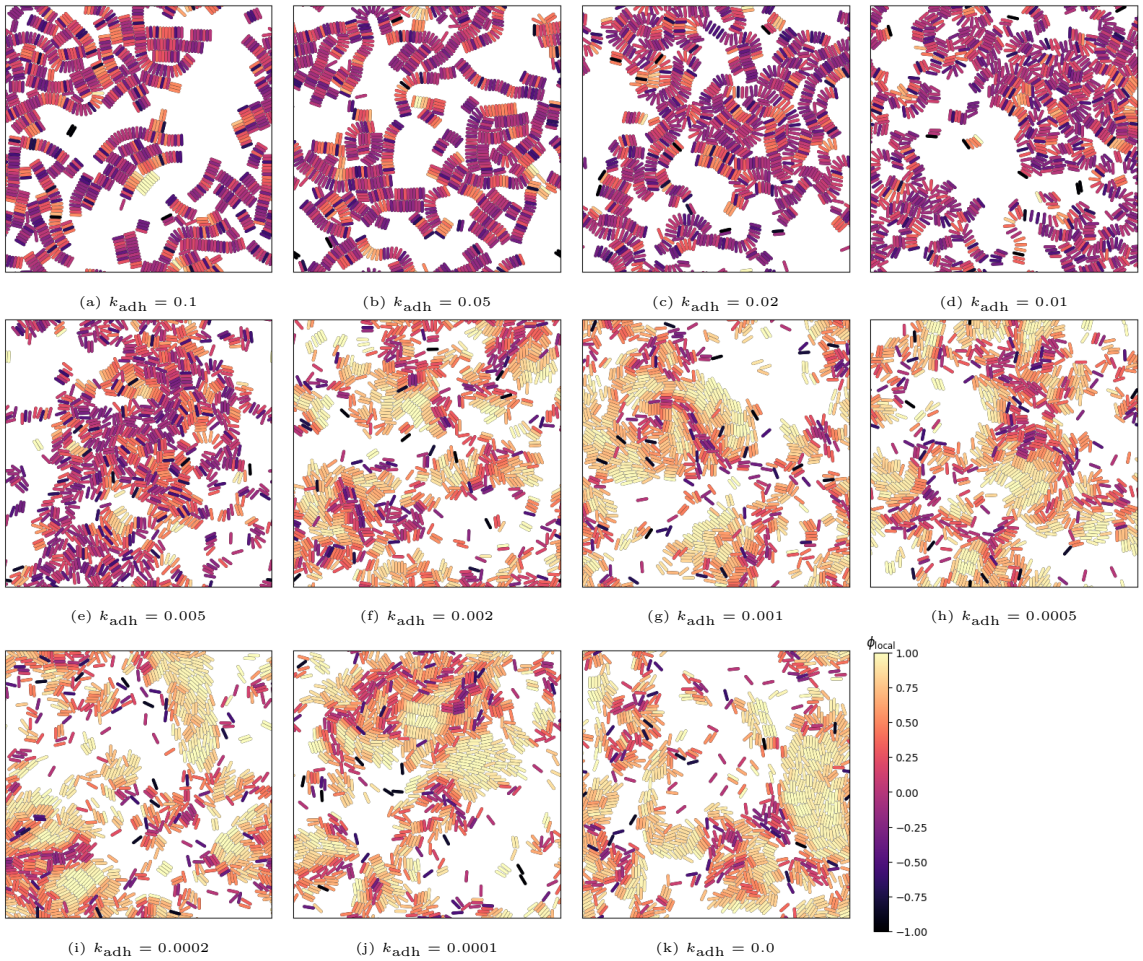


Figure 34: Particle colonies in a 2D system, for different adhesive strengths, coloured according to the individual local order parameter ϕ_{local}, at $t = 1000$, with $N = 1000$, $\rho = 0.5$

5.2 Implementation

Though we would enjoy an immaculate implementation of our model, nothing is perfect, and perfection is only a dream we chase. So let us chase this dream and discuss our envisioned implementation improvements.

5.2.1 Determining the moment of equilibrium

In an effort to filter any transient behaviour in the system, we determined an equilibrium point t_{equi} based on the local order ϕ_{local}. Our current approach to determine t_{equi}, as described in Appendix A.1, can in some cases lead to an inadequate result. The current approach is to smooth the local order parameter over time, determine the point of steepest ascent and look for the moment the inclination

becomes negative after that; this is determined as t_{equi}. If it does not reach this point, it returns the last time step.

There are two problems with this approach. Sometimes the local order increases in two steps, like in Figure 15c for $\rho = 0.75$. In this case, the steepest point could already occur in the first step, making the determined t_{equi} follow shortly after, even though the second step still needs to occur. In this example, the system clearly has not yet reached equilibrium at the determined t_{equi}. A second complication is a reliance on the decrease of the local order; in practice this almost always happens. However, when we look at Figure 15c again, we see that the local order does not decrease after the second increase step. This is partially because local order seemingly only just equilibrated, leaving little time for stochastic effects to push the local order down again. However, one can imagine that lesser fluctuating behaviour, where the local order would not decrease significantly, such that it would be noticeable after smoothing, would pose a problem since the equilibrium would never be detected.

Both problems can be addressed by looking at the extrema in order at the end of the curve with a defined maximum deviation. In this procedure, the algorithm would start at the final time step t_f and walk back in time. It records the minimum and maximum values it encounters and checks if the difference between these is smaller than a certain acceptable difference e. If the difference would exceed this e, then the algorithm stops and returns the current time step as t_{equi}, see appendix A.4 for the basic implementation of this suggestion in Python. The only problem this suggestion poses is that it would consider a system that has never altered in order as equilibrated. However, we can work around this problem by combining it with the existing algorithm, of which the highest value for t_{equi} should be taken as the point of equilibrium.

The additional advantage of this approach would be that we would not be bound to the local order parameter to determine t_{equi}. We could also use the nematic order parameter, which decreases for large tube radii. Furthermore, we could compare the two to determine how good of a measure these quantities are to determine the moment of equilibrium.

5.2.2 Performance

Though the performance of the simulation was significantly better than before, there is still room for further improvement. We already talked about apparent heap allocations and the importance of passing by reference instead of copying an object in section 4.1.2. In this section, we present additional performance improvements for the implementation of the model. The presented optimisations are ordered from most to least impactful while taking in account the implementation effort.

The most impactful improvement would be expanding the Verlet list method to a Verlet box method [44]. In the current method, updating the Verlet list has a complexity of $\mathcal{O}(N^2)$. In the Verlet-box method, the environment is divided into boxes, much larger than the skin radius of the Verlet list, where we keep track of the particles. When we need to update the Verlet list, we only check the surrounding boxes instead of all the particles, reducing the complexity of the Verlet list update.

Another improvement is adding the `constexpr` statement in evaluating the dimension and geometry type. Using this keyword, these are converted into constant expressions, which are evaluated at compile time. Therefore, the program does not have to check the dimension and geometry at every time step during run time, reducing the needed computation. It should be stated that the impact of this improvement is minimal but also requires little effort and is therefore worth implementing.

A larger endeavour would be restructuring the code in a more streamlined manner by writing a single for-loop for all particles, considering all interactions per individual particle. This would enable a smoother parallel implementation, where the computational load would be split over different cores at once instead of multiple times for different interactions. Consequently, the Verlet list should be implemented as an attribute of the Particle structure. Updating the Verlet list would still happen outside of this main particle loop.

The final recommendation we can give to future programmers who wish to improve the program's performance is to use the Clion profiler [45]. We recommend using this profiler in remote mode. Assuming this is a computer cluster running Linux, the profiler uses `perf` to give insight into the

program's bottlenecks. The benefit of remote development is that you are confident that the insights you obtain concern the exact program you want to run.

To illustrate why this is desirable, we want to give the reader an example from our own experience. In this project, we used Very Sleepy CS to profile our for Windows compiled code. Because we only profiled our code locally on Windows, we did not detect that our code initially did not run in parallel, reserving 20 cores on the cluster while using only a single one. A fact we only discovered after a welcome e-mail from Dr. Anton Akhmerov, for whom we are very grateful.

6 Conclusion

The goal in this thesis was to design a 3D simulation of a motile bacterial colony, able to simulate up to 10 000 particles, to investigate how the density and newly introduced adhesion strength affect the dynamics of the system.

6.1 Model

The model has a new drag force corresponding to that of a cylinder, which is a close approximation to that of the sphero-cylindrical particles we have in our simulation. The drag coefficients increase with aspect ratio a. The skin friction and the pressure drag are very similar due to the low Reynolds regime, making the previous model, which used the drag coefficient of a sphere, a surprisingly good estimation, particularly for $a = 4$. We consider the new drag force an improvement over the old one due to its closer corresponding physical foundation and its dependence on aspect ratio.

The new adhesive interaction emulates the EPS using attractive spring forces between two particles reciprocally as well as between particles and solid surfaces. A secondary effect of the adhesive interaction is the inducement for smectic alignment, also observed in nature (Dr. Idema, 2021). For high adhesion spring constants $k_{\text{adh}} \geq 0.01$, we observe long chains of laterally stacked particles, for which we can already empirically state not to correspond with nature. The model does not allow for head to tail adhesion, therefore not holding the chains together.

A suitable value for k_{adh} was found. However, we would advise against $k_{\text{adh}} \geq 0.01$, for any modeller who intends to simulate the physical world, due to the absence of significant local order as well as the formation of laterally stacked chains.

Unfortunately, we overlooked an angular velocity term in our model, disallowing the particle to concurrently move perpendicular to the particle direction $\hat{u}$ as well as rotate in the plane perpendicular to its movement direction $\hat{v}$. This reduces the degrees of freedom from three to two, possibly inducing a slight increase in order. This physicality flaw can be fixed by introducing an additional direction $\hat{w} \equiv \hat{u} \times \hat{v}$. Fortunately, due to the self-propelling nature of the particles as well as the definition of direction $\hat{v}$, the contribution of the additional $\hat{w}$ term is small, making its impact or lack thereof modest at most.

6.2 Implementation & performance

Our goal was simulating 10 000 particles within a reasonable time. With our current implementation, such a simulation with 8000 time steps takes 15 to 65 hours, depending on the geometry and size of the environment. Therefore, we consider our goal achieved. Where the previous code took 19 hours to compute 1000 particles, we are able to do this in 8 minutes, a performance improvement of factor 150.

We achieved this performance using the Verlet list method to keep track of a particle's neighbours by moving the custom vector class from the heap to the stack memory. Additional improvements were enabling -O2 compiler optimisation, parallel computation, and passing object by memory address instead of by copy.

Currently, the main bottleneck of the system performance is the allocation and deallocation of memory for ghost particles at the periodic boundaries. This can be solved by reserving memory at initialisation and reusing the same memory addresses for the potential creation of ghost particles throughout the program.

An additional opportunity for a significant performance increase lies in the dynamic allocation of memory for the JSON object that stores the particles. The dynamic allocation only happens in the first 1000 time steps but can be avoided by allocating memory at the start of the program. A final performance improvement would be expanding the Verlet list method to a Verlet-box method by adding an additional layer on top of the lists, reducing the list update complexity.

The main drawback in our data analysis was the method for determining the point of equilibrium t_{equi}. This method was based on the ascent of the local order parameter ϕ_{local} and assumed a single-step phase transition with stochastic behaviour around an average equilibrated value. However, this

resulted in large deviations from the determined equilibrium in the case of two-step phase transitions. We proposed a new method that relies on the extreme deviations around an equilibrated mean, given an acceptable error. It accounts for two-step phase transitions and can be applied more broadly; for descending parameters as well.

6.3 Results

To characterise the dynamics of the system, we computed the local polar, global polar and nematic order parameters, the mean-squared displacement (MSD), and Reynolds number, for different densities ρ, spring adhesion constants k_{adh}, and tube radii R_{tube} for the tube environment.

For the local order parameter ϕ_{local}, we observed that ρ had a positive effect on the equilibrated ϕ_{local}, though equilibrating also takes more time for higher ρ. For $\rho > 0.75$, the system did not equilibrate within the 8000 simulated time steps. We observe a sigmoid-shaped phase transition from a small to a high ϕ_{local}, though the phase transition occurs in two steps on some occasions.

In the periodic box environment, we found the linear relation $\phi_{\text{local}} = (1.49 \pm 0.09)\rho - 0.56$, indicating that a maximal $\rho = 1$ would also lead to a maximal order $\phi_{\text{local}} = 1$. In the tube environment, we found a positive correlation between tube radius R_{tube} and the ϕ_{local}, with a higher ϕ_{local} than in the periodic box, suggesting there exists an optimal R_{tube} with a maximal ϕ_{local}. The adhesion spring constant k_{adh} did not significantly affect the ϕ_{local}, apart from disallowing high ϕ_{local} for high k_{adh}, due to the clustering of anti-aligned particles with restricted movement.

The global polar order parameter ϕ_{global} stayed nearly zero throughout a simulation and only slightly fluctuated for high densities $\rho \geq 0.75$ and low adhesion spring constants k_{adh}, without any found correlations.

A narrower tube induces a higher nematic order parameter ϕ_{nematic}. Furthermore, there seems to be a relation between ϕ_{nematic} and the density ρ, where low and high ρ relate to higher ϕ_{nematic}, while intermediate $\rho \sim 0.5$ relate to low ϕ_{nematic}. A possible explanation is the combination of a wall effect at low ρ and a clustering effect at high ρ. At low densities, the majority of the particles collide with the tube wall and remain swimming parallel against the wall, unable to escape without any particle-particle collisions. At high densities, on the other hand, the particles are able to form aligning clusters, similar to the case for the positive correlation between ρ and ϕ_{local}. For now, these relations are only hypothetical and should be validated in future research.

When varying the adhesion spring constant k_{adh}, we find that for $k_{\text{adh}} \geq 0.001$, ϕ_{nematic} does not increase, probably due to the inhibition of free movement.

In the first stage after equilibrating, the particles move ballistic, i.e. MSD $\propto t^2$, without any indication of dependency on density, geometry or tube radius. In the last 1000 time steps, the dynamic was more diffusive, i.e. similar to MSD $\propto t$. However, this last result was not averaged over multiple time intervals and therefore less accurate. For $\rho \geq 0.8$, the MSD was significantly lower. This is, however, because these systems had not yet equilibrated.

The Reynolds number $Re \propto 10^{-3}$ is two orders of magnitude larger than the literature predicts for *E. coli* [29]. This difference could be explained by the underestimation of the viscosity η, for which we have not taken in account the secreted EPS and its effect on the viscosity. The higher η would result in a lower Re, due to inverse squared relation $Re \propto \eta^{-2}$. Despite the large discrepancy, our choice to neglect the inertial forces in our equation of motion is still valid because the $Re \ll 1$.

We observed a seeming relation between ρ and Re, where both low and high ρ correlated to higher Re; the latter was to our surprise. A possible explanation is the abundance of aligned clusters at higher densities, where the particles maintain their velocity better, resulting in a higher Re.

As expected, a higher adhesive strength negatively correlated with the Reynolds number, by the intuitive reasoning of 'more stickiness means less mobility'.

7 Outlook

In the introduction, we highlighted the delightful synergy between theoretical and experimental research. It would come as no surprise, seen that this work is of a theoretical nature, that it can still use an experimental touch. The principal opportunity lies in getting a better grasp for the assumed parameters such as the adhesion spring constant k_{adh} both for particle-particle and particle-surface adhesion, the self-propulsion coefficient f_{sp}, the particle-surface friction coefficient μ, and a suitable dynamic viscosity η that is not solely based on water. Furthermore, on a more macro scale, the results found in this thesis could be validated in an experiment with motile non-dividing *E. coli*.

In the current state, the model still has untapped potential. The effect of different aspect ratios is still left untouched, as well as differing the adhesive strengths for the particle-particle and the particle-surface interactions. Furthermore, heterogeneity can be introduced by enabling different particle sizes and self-propulsion forces. These features are already built into the system but left untreated to satisfy the scope with the limited available time.

In addition, we propose some additional features to further extent the model. At the moment, once initialised, the system is still entirely deterministic. Adding randomness to both the size and direction of the propulsion force could mimic a more Brownian motion.

Furthermore, the simulation environments can be expanded as well, e.g. by adjusting the box environment to a rectangular shape with different adjustable sides, giving more freedom to the modeller. Additionally, the introduction of a flowing environment to enable shear experiments could be interesting as well.

A logical step for a more complete picture is to combine our current model for self-propelling particles with a model for growing bacteria already present in the Idema group. As already stated in the introduction, in spite of the fact that the internal structure is predominately shaped by swimming particles, the migration is still dominated by growing bacteria. So, in conclusion, similar to the case of theoretical and experimental research, to truly grasp the dynamics of a bacterial colony, we are in need of synergy.

A Appendix

A.1 Determining the moment of equilibrium

To filter out transient behaviour, we define an equilibrium point t_{equi}, which we can use to start our measurements from. We determine the moment of reaching equilibrium t_{equi} based on the plateauing of the average local order parameter. To do this, we start by smoothing out the local order parameter curve $\phi_{\text{local}}[t]$ using a convolution with the box function $g[t]$

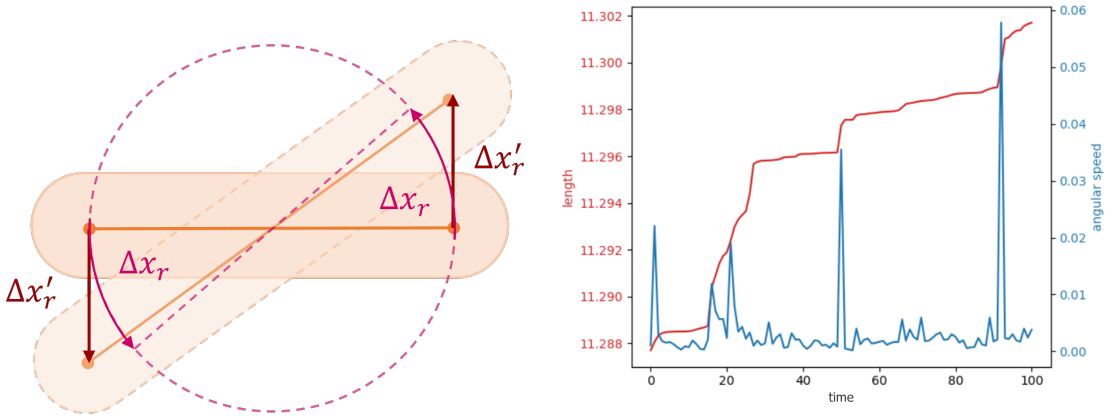
$$(\phi_{\text{local}} * g)[t] = \sum_{\tau=0}^{\tau=t_{\max}} \phi_{\text{local}}[\tau] g[t-\tau], \tag{55}$$

$$g[t] = \begin{cases} 1/\Delta t, & \text{for } 0 \leq t \leq \Delta t \\ 0, & \text{other,} \end{cases} \tag{56}$$

where we take interval Δt as one-hundredth of the total interval. This smoothing step is required due to stochastic nature of the curve, where a sudden increases in ϕ_{local} could lead to a false positive, specifically for determining the steepest point in the curve. Subsequently, we compute the slopes between the points in the middle of every interval Δt. As a last step, we determine the time step with the steepest slope and select the subsequent earliest time step t where the slope becomes negative. We denote this point as the equilibrium point t_{equi}.

A.2 Length increase due to small-angle approximation

The small-angle approximation in eq. 33 results in a small length increase in the particle, which accumulates over time. In section we illustrate the principle behind this and show the accumulating over time in the simulation in Figure 35. To counteract this phenomenon we removed the internal spring which initially existed between the two spheres of the particle and force a length conservation as every calculation step.



(a) An exaggerated schematic representation of the induced length increase, where $\Delta \mathbf{x_r}$ is the theoretical rotational displacement of the particle and $\Delta \mathbf{x_r'}$ is the approximated displacement used in the simulation.

(b) Simulation result showing the length increase as a result of an increase in angular speed for a single particle.

Figure 35: The length increase due to rotation as a result of the small-angle approximation in eq. 33.

A.3 Memory allocation: from the heap to the stack

Initially, the custom vector class allocated memory on the relatively slow heap. We rewrote the code such that the vector objects would be allocated on the faster stack. First, we show the initial vector class `Vector_heap`, using heap memory allocation with the keyword `new`, after which we show the current vector class implemented as a template `VectorT`.

```
struct Vector_heap{

    // ATTRIBUTES
    int length;
    double* value;

    // CONSTRUCTORS
    // Empty
    Vector_heap()
        :   length(0),
            value(nullptr)
    { }

    // Null vector
    Vector_heap(int n)
        :   length(n),
            value(new double[length])       // heap memory allocation with "new"
    {   for(int i = 0; i < length; i++){
            value[i] = 0;
        }
    }

    // Other constructors
    // (..)

    // DESTRUCTOR
    ~Vector_heap(){
        length = 0;
        delete[] value;                 // heap memory deallocation with "delete"
        value = nullptr;
    }
}
```

```
#define Vector VectorT<DIM>     // use VectorT for given dimension (2D/3D)

template<int L>
struct VectorT{

    // ATTRIBUTES
    double value[L]{};          // stack memory allocation
    int length = L;

    // CONSTRUCTORS
    // Null vector
    VectorT(){
        value[L] = {0};
    }

    // Other constructors
    // (..)

    // DESTRUCTOR
    ~VectorT()= default;
}
```

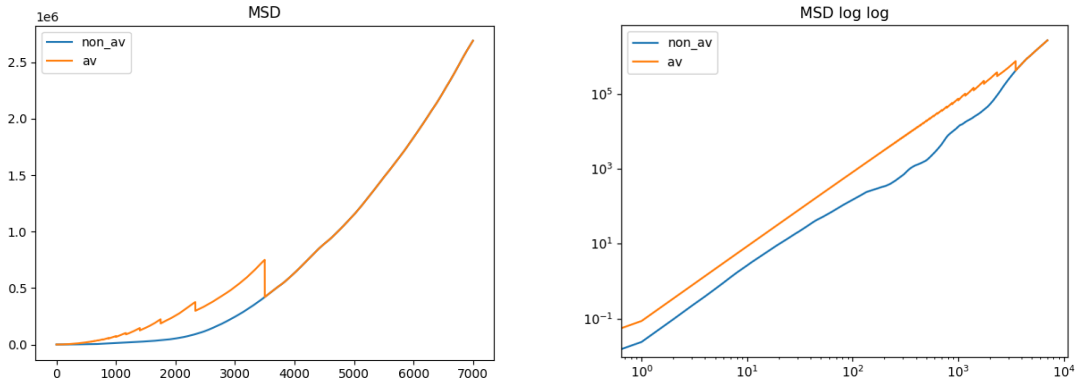
A.4 Suggested improvement for determining equilibrium

```python
def get_t_equi_improved(data: list, e: float) -> int:
    diff, min_value, max_value = 0., 1., 0.
    t = len(data)
    while diff < e:
        t -= 1
        min_value = min(data[t], min_value)
        max_value = max(data[t], max_value)
        diff = max_value - min_value
    return t
```

A.5 MSD sawtooth pattern

The averaging of the MSD over multiple independent equally sized time intervals results in a sawtooth pattern in the tail of the MSD curve on a log-log scale. This is the result of the non-linear relation between the MSD and the time t, combined with the number of time intervals to average the MSD over. Figure 36a shows this particularly well. Here we see a drop in the MSD at $t = 2333$ and $t = 3500$, which corresponds to a third and half the total time. MSD($t = 3500$) is an average over the two time intervals $[t_0, t_{3500}]$ and $[t_{3500}, t_{7000}]$, in which the particles travel further during the second interval, thereby elevating the average. From $t = 3501$ onwards the averaged MSD coincides nicely with the non-averaged MSD, as to be expected since no averaging takes place from this point.

Figure 36b shows us that the averaging is justified when investigating the slope, presenting similar slopes for both cases.



(a) linear scale

(b) log scale

Figure 36: Comparison of the mean-squared displacement MSD as is (blue) and averaged over independent time intervals (orange).

References

[1] Tamás Vicsek et al. "Novel Type of Phase Transition in a System of Self-Driven Particles". In: *Physical Review Letters* 75.6 (Aug. 1995), pp. 1226–1229. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.75.1226. URL: https://link.aps.org/doi/10.1103/PhysRevLett.75.1226.

[2] Eldd Méhes and Tamás Vicsek. "Collective motion of cells: From experiments to models". In: *Integrative Biology (United Kingdom)* 6.9 (2014), pp. 831–854. ISSN: 17579708. DOI: 10.1039/c4ib00115j. arXiv: 1403.1127.

[3] Benoit Ladoux and René Marc Mège. "Mechanobiology of collective cell behaviours". In: *Nature Reviews Molecular Cell Biology* 18.12 (2017), pp. 743–757. ISSN: 14710080. DOI: 10.1038/nrm.2017.98. URL: http://dx.doi.org/10.1038/nrm.2017.98.

[4] M Ballerini et al. "Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study". In: (2008). URL: www.pnas.org/cgi/content/full/.

[5] Ron Sender, Shai Fuchs, and Ron Milo. "Revised Estimates for the Number of Human and Bacteria Cells in the Body". In: (2016). DOI: 10.1371/journal.pbio.1002533. URL: https://erc.europa.eu/funding-and-grants.

[6] C Nicolella. "Wastewater treatment with particulate biofilm reactors". In: *Journal of Biotechnology* 80.1 (June 2000), pp. 1–33. ISSN: 01681656. DOI: 10.1016/S0168-1656(00)00229-7. URL: https://linkinghub.elsevier.com/retrieve/pii/S0168165600002297.

[7] Philip Pearce et al. "Flow-induced symmetry breaking in growing bacterial biofilms". In: *bioRxiv* 1 (2019), pp. 1–6. DOI: 10.1101/627208. arXiv: 627208v1.

[8] Jordi Ferrer, Clara Prats, and Daniel López. "Individual-based Modelling: An Essential Tool for Microbiology". In: *Journal of Biological Physics* 34.1-2 (Apr. 2008), pp. 19–37. ISSN: 0092-0606. DOI: 10.1007/s10867-008-9082-3. URL: http://link.springer.com/10.1007/s10867-008-9082-3.

[9] Judith Pérez-Velázquez, Meltem Gölgeli, and Rodolfo García-Contreras. "Mathematical Modelling of Bacterial Quorum Sensing: A Review". In: 78.8 (2016), pp. 1585–1639. ISSN: 15229602. DOI: 10.1007/s11538-016-0160-6.

[10] Ho Jung Cho et al. "Self-organization in high-density bacterial colonies: Efficient crowd control". In: *PLoS Biology* 5.11 (2007), pp. 2614–2623. ISSN: 15449173. DOI: 10.1371/journal.pbio.0050302.

[11] Waipot Ngamsaad and Suthep Suantai. "Mechanically-driven spreading of bacterial populations". In: *Communications in Nonlinear Science and Numerical Simulation* 35 (June 2016), pp. 88–96. ISSN: 10075704. DOI: 10.1016/j.cnsns.2015.10.026. URL: https://linkinghub.elsevier.com/retrieve/pii/S1007570415003718.

[12] Liyang Xiong et al. "Flower-like patterns in multi-species bacterial colonies". In: *eLife* 9 (2020), pp. 1–27. ISSN: 2050084X. DOI: 10.7554/eLife.48885.

[13] Tomas Storck et al. "Variable cell morphology approach for individual-based modeling of microbial communities". In: *Biophysical Journal* 106.9 (2014), pp. 2037–2048. ISSN: 15420086. DOI: 10.1016/j.bpj.2014.03.015. URL: http://dx.doi.org/10.1016/j.bpj.2014.03.015.

[14] Raimo Hartmann et al. "Emergence of three-dimensional order and structure in growing biofilms". In: *Nature Physics* 15.3 (2019), pp. 251–256. ISSN: 17452481. DOI: 10.1038/s41567-018-0356-9.

[15] Rakesh Sharma et al. "'Unculturable' bacterial diversity: An untapped resource". In: *Current Science* 89.1 (2005), pp. 72–77. ISSN: 00113891.

[16] Tao Long and Dani Or. "Microbial growth on partially saturated rough surfaces: Simulations in idealized roughness networks". In: *Water Resources Research* 43.2 (2007), pp. 1–12. ISSN: 00431397. DOI: 10.1029/2005WR004781.

[17] Steven L. Peck. "Simulation as experiment: A philosophical reassessment for biological modeling". In: *Trends in Ecology and Evolution* 19.10 (2004), pp. 530–534. ISSN: 01695347. DOI: 10.1016/j.tree.2004.07.019.

[18] M. R. Mattei et al. “Continuum and discrete approach in modeling biofilm development and structure: a review”. In: *Journal of Mathematical Biology* 76.4 (Mar. 2018), pp. 945–1003. ISSN: 0303-6812. DOI: 10.1007/s00285-017-1165-y. URL: http://link.springer.com/10.1007/s00285-017-1165-y.

[19] Volker Grimm. “Ten years of individual-based modelling in ecology: What have we learned and what could we learn in the future?” In: *Ecological Modelling* 115.2-3 (1999), pp. 129–148. ISSN: 03043800. DOI: 10.1016/S0304-3800(98)00188-4.

[20] Martin Gardner. “Mathematical Games”. In: *Scientific American* 223.4 (Oct. 1970), pp. 120–123. ISSN: 0036-8733. DOI: 10.1038/scientificamerican1070-120. URL: https://www.scientificamerican.com/article/mathematical-games-1969-01%20https://www.scientificamerican.com/article/mathematical-games-1970-10.

[21] Mya R Warren et al. “Spatiotemporal establishment of dense bacterial colonies growing on hard agar”. In: *eLife* 8 (Mar. 2019). ISSN: 2050084X. DOI: 10.7554/eLife.41093. URL: https://elifesciences.org/articles/41093.

[22] William P.J. Smith et al. “Cell morphology drives spatial patterning in microbial communities”. In: *Proceedings of the National Academy of Sciences of the United States of America* 114.3 (2017), E280–E286. ISSN: 10916490. DOI: 10.1073/pnas.1613007114.

[23] Manon Swinkels. “Collective dynamics of self-propelling rod-shaped particles”. Bachelor’s Thesis. 2019.

[24] Christer Ericson. *Real-time collision detection*. Crc Press, 2004, p. 632.

[25] Dominik Bartuschat et al. “Two computational models for simulating the tumbling motion of elongated particles in fluids”. In: *Computers & Fluids* 127 (Mar. 2016), pp. 17–35. ISSN: 00457930. DOI: 10.1016/j.compfluid.2015.12.010. arXiv: 1503.06869. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793015004077.

[26] M. Mercedes Tirado, Carmen López Martínez, and José García de la Torre. “Comparison of theories for the translational and rotational diffusion coefficients of rod-like macromolecules. Application to short DNA fragments”. In: *The Journal of Chemical Physics* 81.4 (Aug. 1984), pp. 2047–2052. ISSN: 0021-9606. DOI: 10.1063/1.447827. URL: http://aip.scitation.org/doi/10.1063/1.447827.

[27] Tolga Kaya and Hur Koser. “Characterization of Hydrodynamic Surface Interactions of ¡i¿Escherichia coli¡/i¿ Cell Bodies in Shear Flow”. In: *Physical Review Letters* 103.13 (Sept. 2009), p. 138103. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.103.138103. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.138103.

[28] “Collective dynamics in growing bacterial colonies”. Master’s Thesis. 2019.

[29] Tanvir Sajed et al. “ECMDB 2.0: A richer resource for understanding the biochemistry of E. coli”. In: *Nucleic Acids Research* 44.D1 (2016), pp. D495–D501. ISSN: 13624962. DOI: 10.1093/nar/gkv1060.

[30] Deepti Jain et al. “Bacterial Spring Constant in Log-Phase Growth”. In: vol. 1349. 2011, pp. 161–162. ISBN: 9780735409057. DOI: 10.1063/1.3605786. URL: https://doi.org/10.1063/1.3605786%20http://aip.scitation.org/doi/abs/10.1063/1.3605786.

[31] Kenneth Alambra. *Water Viscosity Calculator*. URL: https://www.omnicalculator.com/physics/water-viscosity (visited on 02/01/2022).

[32] Niels Lohmann. *JSON for Modern C++*. Version 3.5.0. URL: https://github.com/nlohmann.

[33] Farzeen Zehra et al. “Comparative Analysis of C++ and Python in Terms of Memory and Time”. In: (Dec. 2020). DOI: 10.20944/PREPRINTS202012.0516.V1. URL: https://www.preprints.org/manuscript/202012.0516/v1.

[34] Loup Verlet. “Computer ”Experiments” on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules”. In: *Physical Review* 159.1 (July 1967), p. 98. ISSN: 0031899X. DOI: 10.1103/PhysRev.159.98. URL: https://journals.aps.org/pr/abstract/10.1103/PhysRev.159.98.

[35] Richard Mitton and Michael Vance Vladimir Panteleev, Richard Munn. *Very Sleepy CS*. 2014. URL: http://www.codersnotes.com/sleepy/.

[36] A Faasse. *Defect dynamics in populations of self-propelling rod-shaped bacteria*. 2020.

[37] Martin Zapotocky, Paul M Goldbart, and Nigel Goldenfeldt. "Kinetics of phase ordering in uniaxial and biaxial nen1atjc fiin1s". In: *PHYSICAL REVIEW E* 51.2 (1995).

[38] Jonathan Naylor et al. "Simbiotics: A Multiscale Integrative Platform for 3D Modeling of Bacterial Populations". In: *ACS Synthetic Biology* 6.7 (July 2017), pp. 1194–1210. ISSN: 2161-5063. DOI: 10.1021/acssynbio.6b00315. URL: https://pubs.acs.org/doi/10.1021/acssynbio.6b00315.

[39] M. C. Bott et al. "Isotropic-nematic transition of self-propelled rods in three dimensions". In: *Physical Review E* 98.1 (July 2018), p. 012601. ISSN: 2470-0045. DOI: 10.1103/PhysRevE.98.012601. arXiv: 1801.08790. URL: https://link.aps.org/doi/10.1103/PhysRevE.98.012601.

[40] M. Cristina Marchetti et al. "Minimal model of active colloids highlights the role of mechanical interactions in controlling the emergent behavior of active matter". In: *Current Opinion in Colloid & Interface Science* 21 (Feb. 2016), pp. 34–43. ISSN: 13590294. DOI: 10.1016/j.cocis.2016.01.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S1359029416300024.

[41] Clemens Bechinger et al. "Active particles in complex and crowded environments". In: *Reviews of Modern Physics* 88.4 (2016). ISSN: 15390756. DOI: 10.1103/RevModPhys.88.045006. arXiv: 1602.00081.

[42] Nicholas C. Darnton et al. "On torque and tumbling in swimming Escherichia coli". In: *Journal of Bacteriology* 189.5 (Mar. 2007), pp. 1756–1764. ISSN: 00219193. DOI: 10.1128/JB.01501-06/FORMAT/EPUB. URL: https://journals.asm.org/journal/jb.

[43] S. Bala Subramanian et al. "Extracellular polymeric substances (EPS) producing bacterial strains of municipal wastewater sludge: Isolation, molecular identification, EPS characterization and performance for sludge settling and dewatering". In: *Water Research* 44.7 (Apr. 2010), pp. 2253–2266. ISSN: 00431354. DOI: 10.1016/j.watres.2009.12.046. URL: http://dx.doi.org/10.1016/j.watres.2009.12.046%20https://linkinghub.elsevier.com/retrieve/pii/S0043135410000059.

[44] Zhenhua Yao et al. "Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method". In: *Computer Physics Communications* 161.1-2 (2004), pp. 27–35. ISSN: 00104655. DOI: 10.1016/j.cpc.2004.04.004. arXiv: 0311055 [physics].

[45] JetBrains s.r.o. *Clion*. Version 2021.3.3. Feb. 1, 2022. URL: https://jetbrains.com/clion/.