Chapter 1

Implementation of interactions in simulations

All the code work in this thesis is built on top of an existing code written and used in Microscale Robotics Lab, which can be found in repository [**sharma_simulations_2023**]. Existing code performed simulations on 2D active Brownian particles moving in open or closed boundary featuring confinement. Original code used an Euler integrator to perform dry simulations of Active Brownian Particles ensembles with the only interaction being the steric hard spheres correction. Moreover, it could simulate variously shaped hard boundaries as well as periodic and open boundary conditions.

1.1 Simulation Framework

In order to simulate most physical systems, some fixed steps are needed and our case study makes no exception. The main nodes of our simulation framework are:

1. Creation and preparation of all needed files

2. Initialization of particle ensemble inside the simulation box

3. Updating loop: a `for` loop on time steps, which includes:

    (a) Force computation at time n, starting from positions at time n
    (b) Generation of the needed random numbers for noise
    (c) Integrator step: one step of the integration algorithm to get the ensemble at time $n + 1$
    (d) Application of boundary conditions and hard spheres correction
    (e) File writing of positions and orientations of all particles in ensemble
    (f) Computation and file writing of time-dependent global properties, e.g. polarization, pair correlation function etc.

4. Files closing

5. Plotting of necessary quantities, saving of animation for dynamics

The main file that gets created at point 1. is the *history* file, the one referred to in (d), to store positions and orientations of all particles. It gets prepared as a comma spaced values (CSV) files, where the column names are written in advance. CSV formatted files are more understandable and universally readable, which is useful when exchanging data between programming languages and manual checks are needed. At every step — or every n_{ds} steps when a down-sampling is used — we append a line to the history file, corresponding to the new simulation instant. In the original simulation code the full history was kept in a vector and then written to the file but in this new version we just update a single instant of the simulation and then write it to the file to improve memory efficiency.

In point 2. particles position are randomly initialized, with a uniform distribution in the simulation space. We need to make sure that particles are not superimposed to avoid nonphysical effects, thus we utilize the hard spheres correction (see section 1.3.2) right after the generation of particles' positions.At this point it is possible to compute forces and torques acting on every particle, which are needed to compute positions at the next step.

From the computational point of view, our simulation framework is based upon object oriented programming: particle ensemble is initialized as an instance of the ABPE class, which holds all the variables that define the state of the system at a certain time 1.1. Ensemble parameters are numbers and all single

Variable	Type	Description
Np	Int64	Number of particles
L	Float64	Size of observation space (µm)
R	Float64	Particle radius (µm)
T	Float64	Temperature (K)
v	Vector{Float64}	Self-Propulsion Velocity ($\mu m\,s^{-1}$)
ω	Vector{Float64}	Self-Propulsion Angular velocity ($rad\,s^{-1}$)
D_T	Float64	Translational diffusion coefficient ($\mu m^2\,s^{-1}$)
D_R	Float64	Rotational diffusion coefficient ($rad^2\,s^{-1}$)
x	Vector{Float64}	Particle X position (µm)
y	Vector{Float64}	Particle Y position (µm)
θ	Vector{Float64}	Particle orientation (rad)

Table 1.1: Structure definition of `ABPE` in Julia

particle variables, such as positions, are vectors of Np elements.

In point 3. positions and orientations get updated with both the deterministic and stochastic parts of the dynamics. This operation can be done either by updating a single instance of ABPE class or by appending to a pre-allocated vector one instance for every time step. The first method is more memory efficient, and it is the one to prefer if no memory is needed, i.e. only step n counts to determine what happens at step $n + 1$. A minimal example of this workflow, with Euler scheme as an integrator, is in algorithm 1.

1.2   Methods

All the simulation and analysis code for this thesis was written in julia [**julia**], except for the machine learning and inference part, where the Graph Network

---

Algorithm 1 The simulation algorithm

1: **for** i in $[1, \texttt{Np}]$ **do**
2:    $\vec{r}_i \leftarrow \vec{r}_{i,0}$ ▷ *position $\vec{r}_{i,0}$ is randomly initialized in the simulation space*
3:    $\theta_i \leftarrow \theta_{i,0}$ ▷ $\theta_{i,0} \sim \text{Uniform}([0, 2\pi])$
4:    $v \leftarrow v_0$ ▷ *Can be initialized as a constant or a random variable*
5: **for** n in timesteps **do**
6:    **for** i in $[1, \texttt{Np}]$ **do**
7:       $\vec{F}_{i,n} = \sum_{j \neq i}^{Np} \vec{F}_{ij,n}$
8:       $w_{i,n} \sim \mathcal{N}(0, \sqrt{2D_t \Delta t})$
9:       $z_{i,n} \sim \mathcal{N}(0, \sqrt{2D_r \Delta t})$
10:       $\vec{r}_{i,n+1} \leftarrow \vec{r_n} + w_{i,n} + v(\cos\theta_{i,n}, \sin\theta_{i,n})\Delta t + \vec{F}_{i,n}\Delta t D_t / k_B T$
11:       $\theta_{i,n+1} \leftarrow \theta_{i,n} + z_{i,n} + \omega_{i,n}\Delta t + M_{i,n}\Delta t D_r / k_B T$

---

definition and training makes use of Python libraries like PyTorch [**pytorch**] and PyTorch-geometric [**pyg**], along with standard scientific computing tools like NumPy [**numpy**] and Pandas [**pandas**].

The Heun algorithm was chosen as the standard integration scheme for all the simulations, and integration steps vary with the different potentials and are indicated case by case. All the simulations were performed with real units, so that particles have a radius of $2\,\mu m$ and the value for solvent viscosity is kept fixed at $1 \times 10^{-3}\,\text{Pa}\,\text{s}$. If not indicated, standard temperature for simulations is held fixed at $300\,\text{K}$.

All simulations were performed in a two-dimensional square environment, where particles are disks, featuring periodic boundary conditions and a non-superposition condition (hard spheres correction). Given that positions are simulated in open periodic boundary, all interactions are calculated periodically too, taking the shortest between the in-box and the periodic distance.

1.3   Boundary Conditions and Hard Spheres Correction

1.3.1   Boundary Conditions

Original code featured open, periodic or hard boundary conditions. We will not go through all of them, but briefly, open boundary conditions means letting particles free to move out the simulation box, which in this case becomes just an *observation space*, while a hard wall involves computing boundary's gradient in any point and using it as a locally straight wall condition.

Periodic boundary conditions for a square simulation area are straightforward to make work if one places the origin in the center of the square. Then, if L is square's side, limits for particles' motion will be $\pm L/2$ in both axes. The actual limit for them to be reflected is $L/2 + R$, meaning a particle must be entirely sticking out before being moved to the other side. These physical boundary conditions are implemented according to algorithm 2.

Algorithm 2 Periodic Boundary Conditions

1: **for all** positions $(x, y)_i$ **do**
2:     **if** $|x_i| > L/2 + R$ **then**   ▷ *R is the particles' radius, L box side*
3:        $x_i \leftarrow x_i - \text{sign}(x_i)L$
4:     **if** $|y_i| > L/2 + R$ **then**   ▷ *R is the particles' radius, L box side*
5:        $y_i \leftarrow y_i - \text{sign}(y_i)L$

1.3.2   Hard Spheres Correction

Original simulation code featured just steric hard spheres correction as interactions among particles, which are enough to study clustering, MIPS and boundary accumulation. The hard spheres correction follows algorithm 3 according to [**callegari_numerical_2019**].

Algorithm 3 The hard spheres correction algorithm

1: **for all** couples of particles $\{i, j\}$ **do**
2:     $d_{i,j} \leftarrow d(\mathbf{r}_i, \mathbf{r}_j)$ ▷ $d(\cdot, \cdot)$ *is the Euclidean distance*
3:     $\mathbf{n}_{i,j} = (\mathbf{r}_i - \mathbf{r}_j)/d_{i,j}$
4:     **if** $d_{i,j} < 2R$ **then**   ▷ *R is the particles' radius*
5:        $\mathbf{r}_i \leftarrow \mathbf{r}_i - \mathbf{n}_{i,j}d_{i,j}/2$
6:        $\mathbf{r}_j \leftarrow \mathbf{r}_i - \mathbf{n}_{j,i}d_{i,j}/2$
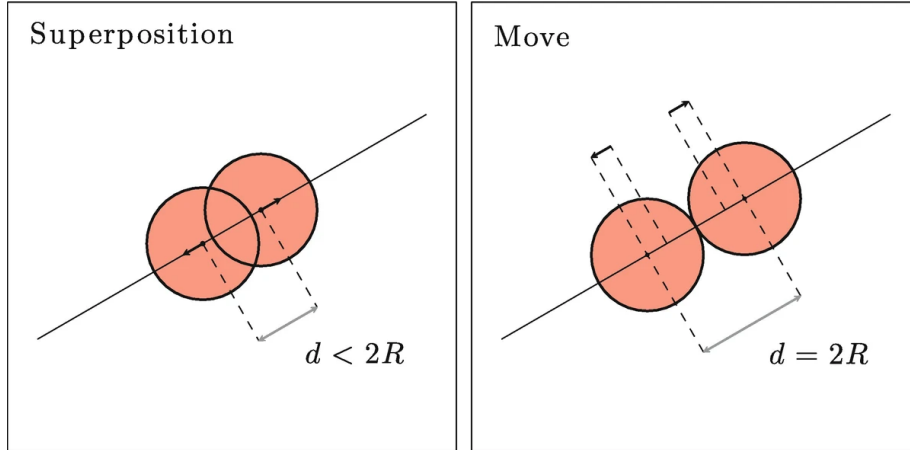

Figure 1.1: If in one step two particles superimpose, hard spheres correction moves them at one diameter distance [**callegari_numerical_2019**]

This operation involves calculating the distance matrix of a set of $\mathtt{N_p}$ particles, i.e. computing $\mathtt{N_p}^2$ distances. Being a $\mathcal{O}(n^2)$ algorithm, this kind of correction can become computationally demanding, especially for high density and velocity systems, where collisions and clustering are more likely to happen. Moreover, correction showed in algorithm 3 is for only one step and it does not guarantee zero superposition afterwards. It becomes important to let the algorithm have a

small tolerance, like 1/1000 of particle radius and to cycle the correction until the number of superposition becomes zero.

In the work cited in section **??**, **martin-gomez_collective_2018** use a steep repulsive potential $\sim r^{-12}$ to simulate the excluded volume. In principle, since it does not require cycling, this approach seems faster, especially if one has to compute interactions. During the development of this thesis some tests were run and, with all integration algorithms, even though computing the excluded volume effect is actually faster than applying a hard spheres correction, the steep potentials involved require a much smaller time step (1 or 2 orders of magnitude), making the simulation slower.

Separating superimposing particles in a way to keep them in contact may seem a strong assumption, since it is simulating a perfectly inelastic collision. This choice arise due to the finite contact time and micron sized physics of the active particles under study. In macroscopic Newtonian dynamics scenario (balls moving on a pool table), it is relatively easy to state collisions' elastic or inelastic nature. Furthermore, in the Microscale Robotics Lab and in literature [**singh_pair_2024**], Janus active particles collisions are observed to have a finite contact time $\sim$ s. This long collision time encouraged us to implement inelastic collision in simulations.

1.3.3 Modified Periodic Boundary Steric Interactions

Since hard spheres corrections were implemented to study ABPs in confinement rather than open periodic boundary, this interaction could not take periodicity into account leading to possible superpositions at boundaries.

When boundary conditions are periodic, not taking them into account in steric interactions may cause some problems, on top of being not formally correct. With the hard spheres correction as it was described in section 1.3.2, particles are separated when their distance is less than their diameter. With periodic boundary conditions, a particles is reflected *e.g.*, on the left hand side only when its center is out of the right hand side by more than one particle radius (R). This may leave a *frame* of size R around the boundary where inter-particle periodic distance can be less then $2R$.

When combined with potentials that feature a *soft* non-superposition condition, like a diverging positive force at a $2R$ distance, this can lead to pairs of particles sticking out of the border on opposite sides and superimposing a lot before one of them gets reflected on the other side, but when this happens, the force between them diverges, resulting in nonphysical situations like particles moving far outside the box.

To address this issue, we propose a modified version of hard spheres corrections where distances are not computed in a Euclidean fashion, but x and y components are separately checked. For a box of size L, we compute $\Delta x = min\{|x_1 - x_2|, |x_1 - x_2| - L\}$, and the same for y, which is the right periodic distance between any given pair of particles. If $\Delta x < 2R$, particles are guaranteed to superimpose when periodic boundary condition is applied. To eliminate this superposition, we implemented the hard spheres correction as in section 1.3.2 to superimposing pairs at the boundary. This method allows us to identify superpositions at the boundaries, thus giving the possibility to form clusters starting from a *nucleation site* on the border.
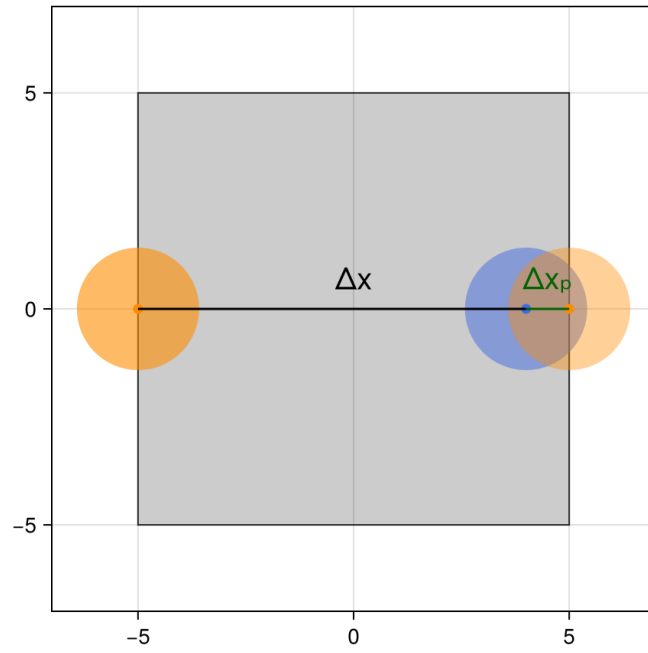
Figure 1.2: Orange particle is sticking out on the left side. Its *real* distance from blue particle on x-axis is Δx. Its periodic projection (faded orange) has a distance from blue particle $\Delta x_p < 2R$, hence superimposing.

1.4 Interaction Forces

1.4.1 Central Potentials

Spring, Coulomb, Lennard Jones, Yukawa, Weeks-Chandler-Anderson are just few examples of the myriad of central potentials that are used to model interactions in physics. The central potential is the basis of this thesis too, and having a fast implementation is crucial to simulate interactions in dense systems.

The first step to apply a potential is to compute a distance matrix between the coordinates of all particles. Then a function $F(d)$, only of inter-particle distance d, is applied to all the entries of this matrix to get the magnitude of the force between each pair of particles. In 2D, radial direction for the force can be computed from distance matrices both for x and y coordinates. If $d_{i,j}^x = x_i - x_j$ and $d_{i,j}^y = y_i - y_j$, then Euclidean distance is $d_{i,j} = \sqrt{(d_{i,j}^x)^2 + (d_{i,j}^y)^2}$, and if $\varphi_{i,j}$ is the radial direction from particle i to particle j, then

$$\begin{cases} \cos \varphi_{i,j} = d_{i,j}^x/d_{i,j} \\ \sin \varphi_{i,j} = d_{i,j}^y/d_{i,j} \end{cases} \tag{1.1}$$

splitting the force in its two components. Now the computed force can be used to calculate the deterministic step in the coordinate as in algorithm 1: given that we are in low Reynolds number regime, *velocity* is proportional to force .

1.4.2 Tested Potentials

We give here the expressions and parameters explanation for the potentials tested in this thesis.

Spring potential

We used this potential to test a strong long-range interaction between particles, with some control over the equilibrium position. Spring force has two parameters, elastic constant k_S and rest length r_0, so that

$$V(r) = \frac{1}{2} k_S (r - r_0)^2 \rightarrow F(r) = -k_S (r - r_0). \tag{1.2}$$

Coulomb potential

What we will often call Coulomb force is actually a r^{-2} order potential with a constant k_C that does not involve charges and it is present only to adjust the strength of interaction, that writes

$$V(r) = -\frac{k_C}{r} \rightarrow F(r) = \frac{k_C}{r^2} \tag{1.3}$$

and any sign variation, as well as the difference between repulsive and attractive, is absorbed in k_C. Being an attractive or repulsive-only potential, it can be used to study the effect of a potential in a relatively simple situation, especially when dealing with aligning interactions. Its divergence is not so steep thus letting us utilize larger integration steps.

Lennard Jones potential

The last interaction potential is Lennard Jones (LJ)

$$V(r) = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \rightarrow F(r) = 24\varepsilon \frac{1}{r} \left[2 \left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \quad (1.4)$$

where σ is a location parameter that in our simulations will be fixed at the value of particle's diameter $2R$, and ε adjusts the strength of interaction. This potential has an absolute minimum for $r = 2^{1/6}\sigma$. LJ is a standard in molecular dynamics and disordered systems, both its original version and its variations are often used in ABP research, since this potential encloses both short range repulsion to avoid superpositions and medium range attraction to give structure to the system. The short range divergence is pretty steep and its strong non-linearity requires very short integration times, better if coupled with advanced integration algorithms.
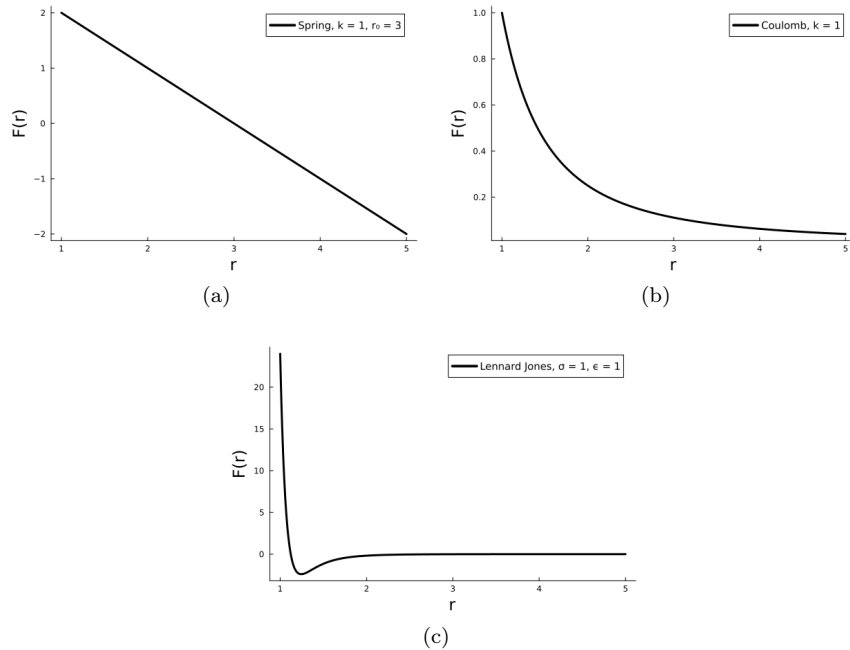

Figure 1.3: Tested interaction forces for given values of their parameters.

1.4.3 Interaction Range

In some situations is useful to have a way to restrict the force computation only to particles closer than a threshold distance.

One could do this to imitate a topological interaction at short range, where a particle only interacts with its nearest neighbors. In the case of circular particles (or spherical in 3D), setting the range to $3R$ is enough to get this effect, since two particles in contact will have their centers at $2R$ and no other particle can be at less than $3R$ in the same directions. This way, central particle will interact with

figura dei 3 potenziali un po' piccola (soprattutto il testo), e forse sarebbe meglio mettere le tre curve nello stesso grafico (se resci a dargli range di forze simile)

perché le figure de grafici in png e non in pdf / svg o altri formati vettoriali?

è vero che abbiamo detto che i grafici sono più belli senza griglia, ma sarebbe utile avere almeno una linea orizzontale a zero, o spostare l'asse orizzontale in modo che passi

6 other particles at most (this is the number of nearest neighbors for hexagonal close packing order 2D) giving the effect of a nearest neighbor interaction in the case of contact. Clearly, this is not a *true* topological interaction, where two nearest neighbors interact regardless of their distance, but it can give similar effects if applied in high packing fraction cases, where particles come in contact.

Moreover, when dealing with fast decreasing potentials, the magnitude of the force between two distant particles can be effectively negligible and even lower than the numerical precision. If this is the case, it is obviously pointless to look into pairs of particles which are more than a certain distance apart. One can compute this distance for any given potential, provided some reference magnitude. An example is given in Figure 1.4. This is of paramount importance
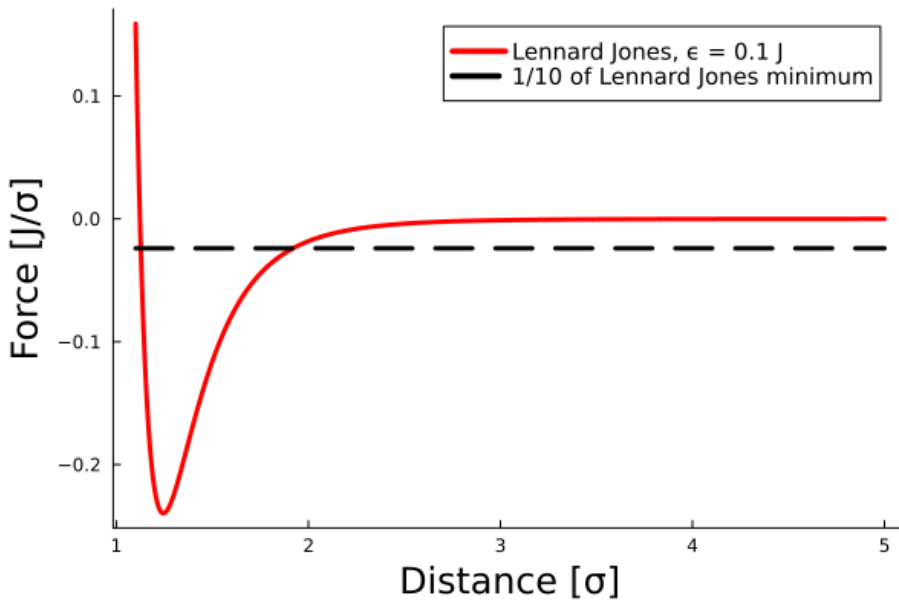

Figure 1.4: Cutoff for Lennard Jones force at $1/10$ of the minimum. The cutoff distance is $\sim 1.9\sigma$

for speed. While a complete calculation of all the possible forces between n particles is an $\mathcal{O}(n^2)$ algorithm (the actual number of possible interactions is $n(n-1)$), the number of particles in a box of given area is constant in n if density remains fixed, so, scaling up the number of particles in the same conditions will be less demanding in terms of computational power.

Inserting a range in interactions makes straightforward to apply them in a periodic fashion. Though interactions were not featured in original code, functions for boundary conditions were implemented in a fast vectorized way, letting us reuse the existing function to reflect particles in periodic boundary conditions (pseudo code in algorithm 2) in the interactions part. The idea is: take one particle and make it the *central* one, then shift coordinates of the other particles so that the *central* is in $(0,0)$, now just apply periodic boundary conditions to the shifted coordinates. This process should guarantee that the *central* particle interacts only once with any of the others, following a simple rule: when choosing between the true particle and the periodically shifted one

se confronti la dimensione del font di questa figura (cutoff) con quello della figura precedente (potenziali), noterai che sono diversi. sarebbe meglio uniformarli, come discutevamo l'altro giorno

the interacting particle is the closest one. An explanation of this process can be found in Figure 1.5.
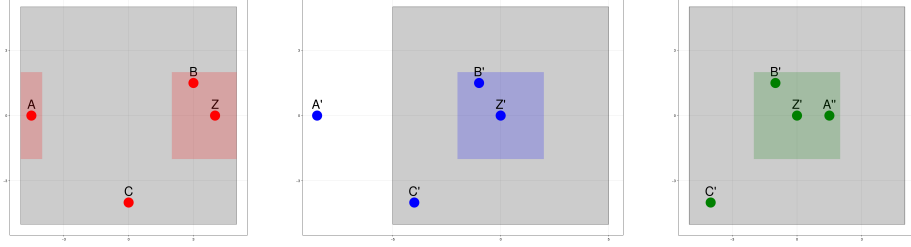

Figure 1.5: Red: original points with Z as central point and its interaction box sticking out on the opposite side. Blue: shifted points w.r.t. Z. Green: shifted points with PBC applied.

i numeri sugli assi della figura su periodic interaction sono illeggibili

1.4.4   Aligning Interactions

There are several examples in literature of aligning interactions applied to ABPs systems [**martin-gomez_collective_2018**, **callegari_numerical_2019**]. Most of them involve applying a torque which explicitly depends on the difference between the orientation angle of the two interacting particles; some of them have a constant torque which is applied to particles closer than a certain distance. To the best of our knowledge, nobody have implemented interactions in a way that unifies the central force interactions and the aligning interactions.

The model used in this thesis aims to keep a minimal amount of parameters while coupling positions and orientations of interacting particles, actually using only one more parameter than a force-only model. When applying a central potential interaction to a set of particles, the distances r between particles are calculated with respect to the particles' centers and the computed interaction force $F^i(r)$ is applied to the particles centers as $\begin{bmatrix} F_x^i & F_y^i \end{bmatrix}$ (2D simulations). Here every particle is identified by two positions: its center of mass, which lies in its geometric center, and an interaction center, which is translated with respect to the center of mass of an amount αR in the direction of self-propulsion, so that the off center position of each particle writes:

$$\vec{x}_{oc} = \vec{x}_c + \alpha R \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \tag{1.5}$$

where θ is the self-propulsion direction, R is the particle's radius, and $-1 < \alpha < 1$. Compared to the self-propulsion direction, the off-center position is on the front half of the particle when $\alpha > 0$ and on the back half when $\alpha < 0$. Distances, and hence forces, are thus computed between the off-center positions. Applying forces off center results also in an effective torque applied to the particle, leading to both translational (repulsive/attractive) and rotational (aligning) effects, coupling the two degrees of freedom.

The idea is to take into account the underlying particle's asymmetry not only in terms of self-propulsion but also in terms of interactions. It is a fact that the two faces of a Janus particle have different physical and chemical properties, which make them interact differently depending on the angle between their

orientations and the line connecting their centers [**singh_pair_2024**]. This can be due to a plethora of effects like chemical gradients, electrical forces, hydrodynamics or fabrication defects that would be hard to model in details and effectively impossible to simulate for even a few particles. This model takes only the asymmetry and uses it along with standard potentials, to create a minimal dry framework that mimics such a complex wet dynamics. In doing so, we also show that this model displays some interesting statistical mechanics properties.
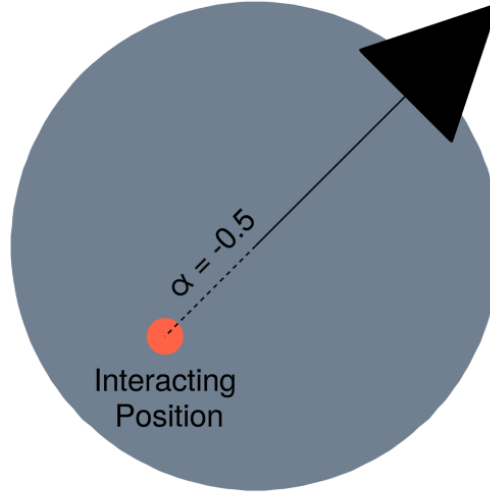

Figure 1.6: Red point: position from which interaction distances are calculated and to which force is applied. In this case, interacting position is at $-0.5R$ from particle's center, *i.e.* on the back side compared to the direction of self propulsion.

1.5 Qualitative Model Validation

Since the real interaction forces and torques between self-diffusio-phoretic particles are still a matter of deep investigation, it is not straightforward to check if a model is accurate or complete. Here, we look into a qualitative comparison between the simulated behaviors and the observed ones, focusing in this case on interactions between pairs or small groups of particles.

It is important to note that often in experiments clusters of particles are not caused by self-assembly, but are there as a by-product of the fabrication process, during which particles may remain attached. Still, observing how single particles interact with clusters gives some insights on contact behaviors, making particle clumps' presence useful. In principle, it is difficult to create stable clusters in simulations, since our framework does not involve the possibility

of physically attached particle, but an attractive potential is needed to mimic observed behaviors.

Due to differences and irregularities in the fabrication process, not all particles have the same self-propulsion velocity. In simulations we can reproduce this generating particles' velocity from a distribution. Although particles orientation is difficult to measure in our experimental videos, it is possible to note that some particles are spinning around and some others have both a linear and a rotational self-propulsion, in a Chiral Brownian Particles fashion, [**callegari_numerical_2019**, **bechinger_active_2016**].

With all of this said, the ingredients for a simulation that reproduces experimental behaviors are: a short-range potential with an attractive part, a linear and angular velocity distribution. We chose Lennard Jones for its short range attraction, keeping $\sigma = 2R$ with R particle's radius, with negative off-center parameter α. Although most videos do not contain enough particles to build a sample velocity distribution, it is noticeable how very fast particles are rare if compared with slow particles. To catch this asimmetry, we generated simulated particles' velocities from an exponential distribution; this is done to maximize the number of collisions as well. Since we do not observe a preferred direction in self-propulsive rotation, particles' angular velocities are generated from a distribution which is symmetric around 0; moreover, few particles present large angular velocities, so we generated them in simulations using a Normal distribution, to have control on its broadness.