Chapter 1

Implementation of interactions in simulations

1.1 Model

There are several examples in literature of aligning interactions applied to ABPs systems. 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 tried before to implement an interaction in the system which unifies the central force and the torque.

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.

If one was to apply a central force to a set of particles, the way would be computing a distance matrix between the positions of particles' centers, calculating the absolute value of the force with a function $F(r)$, and applying it to particle positions, splitting the two components (we are simulating in 2D).

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:

$$\vec{x}_{oc} = \vec{x}_{cm} + \alpha R \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \tag{1.1}$$

where θ is the self-propulsion direction, R is the particle's radius, and $-1 < \alpha < 1$. This way distances, and hence forces, are computed starting from off center positions; applying forces to such positions results both in a translation effect and in a torque applied to the particle, coupling the two degrees of freedom.

The idea is to take into account the underlying particle's asymmetry in simulations. 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 or hydrodynamics, but this model takes only the

asymmetry and uses it along with standard potentials, to create a minimal dry framework that mimics wet dynamics. In doing so, we also show that this model displays some interesting statistical mechanics properties.

1.2 Methods

All the simulation and analysis code for this thesis was written in julia, except for the machine learning and inference part, where the Graph Network definition and training makes use of Python libraries like PyTorch and PyTorch-geometric, along with standard scientific computing tools like NumPy and Pandas.

Heun was chosen as the standard integration scheme for all the simulations, 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 μm and the value for solvent viscosity is kept fixed at $\eta = 10^{-3}$ P.

All simulations were performed in a 2 dimensional square environment, where particles are disks, featuring periodic boundary conditions and a non-superposition condition (hard sphere 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 Previous Developments

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. The interactions among these particles were steric hard sphere correction, which are enough to study clustering, MIPS and boundary accumulation. The hard sphere correction follows algorithm 1 according to [**callegari_numerical_2019**].

Algorithm 1 The hard sphere correction algorithm

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

This operation involves calculating the distance matrix of a set of n particles, i.e. computing n^2 distances. Being a $O(n^2)$ 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.

Since hard sphere 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 the simulation box borders.

Finally, let us explain the physics behind this kind of correction. Moving superimposing particles of just the right amount to keep them in contact may

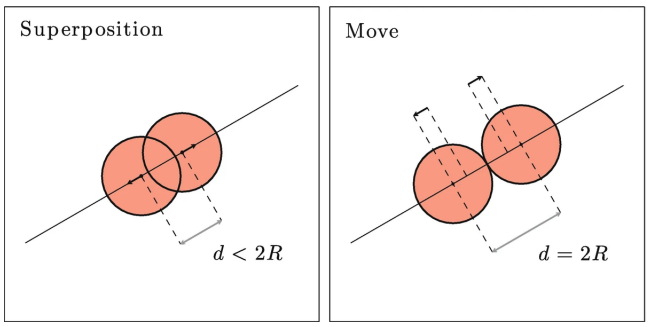

Figure 1.1: [**callegari_numerical_2019**]

seem a somehow strong assumption, since it is simulating a perfectly inelastic collision. Actually, if one wanted to simulate balls moving on a pool table, where macroscopic Newtonian dynamics take place, it would be easy to state whether collisions are elastic or not (in the mentioned case, yes). Here we can exploit what was found by, among others **singh_pair_2024** in [**singh_pair_2024**], where the contact time for Janus colloids was characterized, in different collision scenarios. They found contact times of the order of $\sim$ s, similarly to what is often observed in Microscale Robotics Lab, motivating the choice of a completely inelastic collision, where adhesion forces keep particles together after the impact.

1.4 Periodic Steric Interactions

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 sphere correction as it was described before, particles are moved only when their distance is less than a diameter. With periodic boundary conditions, a particles is reflected e.g. on the left side only when its center of mass is out of the right border of more than a radius, meaning the particle must be entirely out before being reflected. If not taken into account, this may leave a *frame* of size R around the box where particles can superimpose.

In the new version of this correction, distances are not computed in a Euclidean fashion, but x and y components are separately checked, taking $\Delta x = min\{|x_1 - x_2|, |x_1 - x_2| - L\}$, and the same for y, where L is the box size. If one of the two particles is coming out of the box, but it has not been reflected yet, and the other is at the boundary, their difference in one component is larger than L. This way it is possible to identify superpositions at the borders, giving the possibility to form cluster starting from a *nucleation site* on the boundary.

Another, more serious consequence is that, when working with potentials that feature a *soft* non-superposition condition, like a diverging positive force at a $2R$ distance, two particles sticking out of the border on opposite sides may superimpose a lot before one of them gets reflected on the other side, but when this happens, the force between them diverges, leading to instabilities in the simulation.

1.5 All to All 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 large systems of particles.

The first step is to compute a distance matrix between the coordinates of all particles. Then a function $F(r)$ is applied to all the entries of this matrix to get the magnitude of the force between each pair of particles. We are simulating in 2D, so one way to get the radial direction for the force is to compute distance matrices both for x and y coordinates and then dividing them by the Euclidean distance matrix element-wise to get sine and cosine of the radial directions angle. Now the force is split in its two components and it can simply be added to the deterministic step in the coordinate as in algorithm **??**, given that we are in low Reynolds number regime.

1.6 Range

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

One could do this e.g. to

1.7 Aligning Interactions