Active Brownian Particles

Ankush Checkervarty

A dissertation submitted for the partial fulfilment
of BS-MS dual degree in Science

Indian Institute of Science Education and Research Mohali
April 2014

Certificate of Examination

This is to certify that the dissertation titled **Active Brownian Particles** submitted by **Ankush Checkervarty** (Reg. No. MS09018) for the partial fulfillment of BS-MS dual degree programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Ramandeep S.Johal Dr. Rajeev Kapri Dr.Abhishek Chaudhuri
(Supervisor)

Dated: April 25, 2014

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Dr. Abhishek Chaudhuri at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Ankush Checkervarty

Dated: April 25, 2014

In my capacity as the supervisor of the candidates project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Dr. Abhishek Chaudhuri
(Supervisor)

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Dr. Abhishek Chaudhuri for the continuous support of my project and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my final year thesis study.

Besides my advisor, I would also like to thank all my batchmates and friends, for their fruitful discussions whenever i needed them and creating a studious environment.

List of Figures

Abstract

Using simulations, we study the theoretical models of the collective dynamics of the active particles. These particles constitute a large class of non-equilibrium systems where each individual takes energy from the environment and moves depending upon interaction with their neighboring units. We first analyzed the models of polar active particles for both constant speed and variable speed models. This class is generally characterized by two phases- ordered state and disordered state. We explore the phase diagrams using noise intensity and packing fraction as control parameters. We found that transitions occurs from the ordered to disordered state with the increase in noise intensity. We also studied self propelled soft repulsive disks in two dimensions similarly for constant speed and variable speed. These particles have excluded volume interactions and are subject to only rotational noise, but without any aligning interaction. This system shows a clustered state above a critical density and self propulsion speed. We investigate the phase diagrams under these control parameters.

Contents

Chapter 1

Introduction

Active Brownian particles (Fig 1.1) are the particles which can take up energy from the environment, may store it and then convert this energy into systematic movement. Examples of active Brownian particles include moving cells[6],fishes,[8] insects or even non-living matter such as shaken metallic rods [5] and nano-swimmers. These systems show very interesting

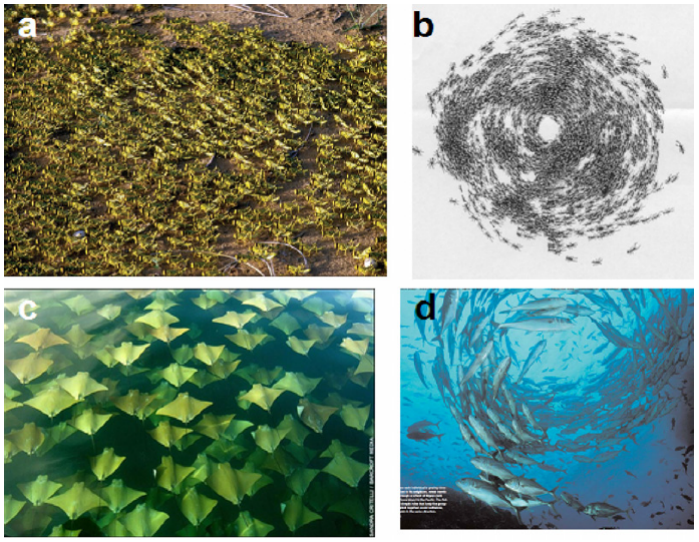

Figure 1.1: (a) Wingless Locusts (b) Rotating colony of army ants (c) A 3D array of Golden rays (d) Fishes (Reproduced from [1])

patterns in their swarms or collective motion like abrupt formation of uniformly moving group from totally random moving individuals or vice-versa, swirling motion : abrupt formation of vortices in the flock, etc. The aim of our study is find out how such collective dynamics emerges from such random motion of individuals units. Before studying the active collective dynamics, we first look at the passive random motion of the individual units.

1.1 Passive Brownian Motion

When a tiny particle is put in a fluid it does random motion which is called Brownian Motion[2]. To model the motion of a single Brownian Particle in 1D, two kinds of forces are basically considered to be controlling the particles motion. First, there is a force which comes from its interaction with the medium. This force is the frictional force($-\zeta v$) proportional to the velocity of the particle. The second one is random force due random collision with other particles in the fluid. The frequency of these collisions is very high. In order to model these random forces we use Gaussian white noise.

$$\langle \delta F(t) \rangle = 0, \quad \langle \delta F(t) \delta F(t') \rangle = 2B\delta(t - t'); \tag{1.1}$$

The first equation attributing the property of force which means it's random hence the average over time is zero, the second equation explains the no-memory property of these forces as the forces at two different times can't have any correlation; the delta function confirms that these correlation follow the same property.

Therefore, the equation of motion for a passive Brownian Particle is

$$m\frac{dv}{dt} = -\zeta v + \delta F(t) \tag{1.2}$$

This is the Langevin Equation. Solving this equation involves easy methods of solving differential equations of the form $dy/dx + P(x)y = Q(x)$, which involves finding integration factor. Multiplying and integrating gives

$$v(t) = v(0)e^{-\zeta t/m} + e^{-\zeta t/m} \int_0^t dt e^{\zeta t'/m} \frac{\delta F(t')}{m}. \tag{1.3}$$

Now in order to find the mean squared velocity we should find $\langle (v(t))^2 \rangle$

$$\langle (v(t))^2 \rangle = v(0)^2 e^{-2\zeta t/m} + \frac{B}{\zeta m}(1 - e^{-2\zeta t/m}) \tag{1.4}$$

In the long time limit the above approaches $B/\zeta m$. But mean square velocity should be kT/m in equilibrium by equipartition theorem. This implies

$$B = \zeta kT. \tag{1.5}$$

This relation is known as Fluctuation-dissipation theorem. This relation basically connects magnitude of fluctuation i.e. B with the strength of dissipation ζ.

1.2 Correlation function

1.2.1 Time correlation function

These functions play an important role in understanding the phenomena in non- equilibrium statistical mechanics. Suppose there is quantity whose A(t) whose correlation function we have to find. First we average this quantity over a time interval we need to find this function

$$\langle A\rangle = \frac{1}{\tau}\int_0^\tau A(t)dt \tag{1.6}$$

Now the fluctuation δA is given by $\delta A = A(t) - \langle A\rangle$ This function δA are generally correlated at different times and this correlation can be measured by the quantity

$$C(t) = \frac{1}{\tau}\int_0^\tau ds\delta A(s)\delta A(t+s). \tag{1.7}$$

1.2.2 Velocity correlation function

This simple time correlation is important as it's related to diffusion constant of Brownian motion. An easy way to understand this correlation is through 1-dimensional diffusion equation.

$$\frac{\partial C(x,t)}{\partial t} = D\frac{\partial^2 C(x,t)}{\partial x^2} \tag{1.8}$$

This can be related to mean square displacement $\langle x^2\rangle$

$$\frac{\partial\langle x^2\rangle}{\partial t} = \int dx x^2\frac{\partial C(x,t)}{\partial t} \tag{1.9}$$

using above two equations and integration by parts

$$\frac{\partial\langle x^2\rangle}{\partial t} = 2D\int dx C(x,t) = 2D \tag{1.10}$$

where D is diffusion constant. But now position (x) and mean square velocity $\langle x^2\rangle$ can be calculated through velocity too.

$$x(t) = \int v(s)ds \tag{1.11}$$

$$\langle x^2\rangle = \langle\int_0^t ds_1 v(s_1)\int_0^t ds_2 v(s_2)\rangle = \int_0^t ds_1\int_0^t ds_2\langle v(s_1)v(s_2)\rangle \tag{1.12}$$

where $\langle\rangle$ represents ensemble average. Taking the time derivative

$$\frac{\partial\langle x^2\rangle}{\partial t} = 2\int_0^t ds\langle v(s)v(t)\rangle \tag{1.13}$$

The term of right hand side inside integral is velocity correlation function. This relation is important as it can be used in both experiment and simulation in calculation diffusion constant, which is a crucial quantity to characterize a Brownian motion.

1.3 Diffusion coefficient and Temperature

Another application of this relation is that it can be used to find out the relation between diffusion constant and temperature, as the diffusion constant is directly proportional to the temperature. Again the displacement can be written as

$$\Delta x(t) = \int_0^t dt' v(t'). \tag{1.14}$$

and using velocity relation Eq. 1.3

$$v(t) = v(0)e^{-\zeta t/m} + \int_0^t dt' e^{-\zeta(t-t')/m} \delta F(t')/m. \tag{1.15}$$

Using same relation Eq 1.1 and doing the averages the we get

$$\langle \Delta(x(t))^2 \rangle = \frac{2kT}{\gamma} \left[t - \frac{m}{\gamma} + \frac{m}{\gamma} e^{-\zeta t/m} \right] \tag{1.16}$$

one can easily see at short times the mean square displacement is quadratic function of time, this is due to effects of initial velocity. But at the larger times noise dominates and relation becomes linear in time.

$$\langle \Delta x^2 \rangle = 2\frac{kTt}{\zeta} \tag{1.17}$$

where ζ is coefficient viscosity. Hence from Eq 1.10

$$D = kT\gamma. \tag{1.18}$$

Before doing molecular dynamics in presence of Langevin heat bath Eq 1.2, I performed molecular dynamics of N particles in the constant NVE ensemble.

1.4 Molecular dynamics in the constant NVE ensemble

In the constant NVE ensemble, we start with $N = 108$ particles in three dimensions. The particles are initially arranged on a cubic lattice and the density is $\rho = 0.8442$ [4]. The initial velocities are chosen randomly and are rescaled to keep the initial temperature as

$T = 0.728$. The interaction between particles is modeled as a Lennard Jones potential :

$$V(r) = 4\epsilon \left(\frac{1}{r^{12}} - \frac{1}{r^6} \right). \tag{1.19}$$

The force between particles is calculated from this potential. The Lennard Jones potential has an attractive part $(1/r^6)$ which is long ranged and a repulsive part $(1/r^{12})$. There is a potential well at ϵ. We use periodic boundary conditions. The coordinate and velocity

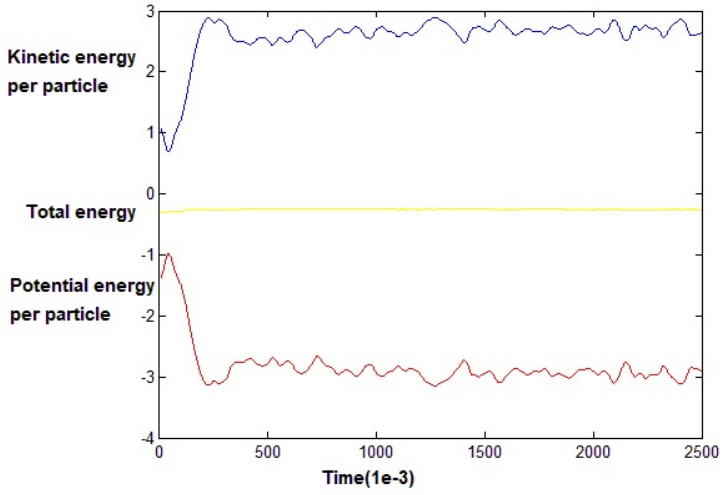

Figure 1.2: Kinetic energy Vs time

updates are done using Verlet's algorithm. The Taylor series of coordinate after a time Δt can be written as

$$r(t + \Delta t) = r(t) + v(t)(\Delta t) + (f(t)/2m)(\Delta t)^2 + O((\Delta)^3), \tag{1.20}$$

where $f(t) = m\ddot{r}$ by Newton's second law. Similarly

$$r(t - \Delta t) = r(t) - v(t)(\Delta t) + (f(t)/2m)(\Delta t)^2 + O((\Delta)^3), \tag{1.21}$$

therefore, in coordinate updates we can sum Eq. 1.20 and Eq 1.21 and ignoring the higher order terms

$$r(t + \Delta t) = 2r(t)r(t - \Delta t) + (f(t)/m)(\Delta t)^2 \tag{1.22}$$

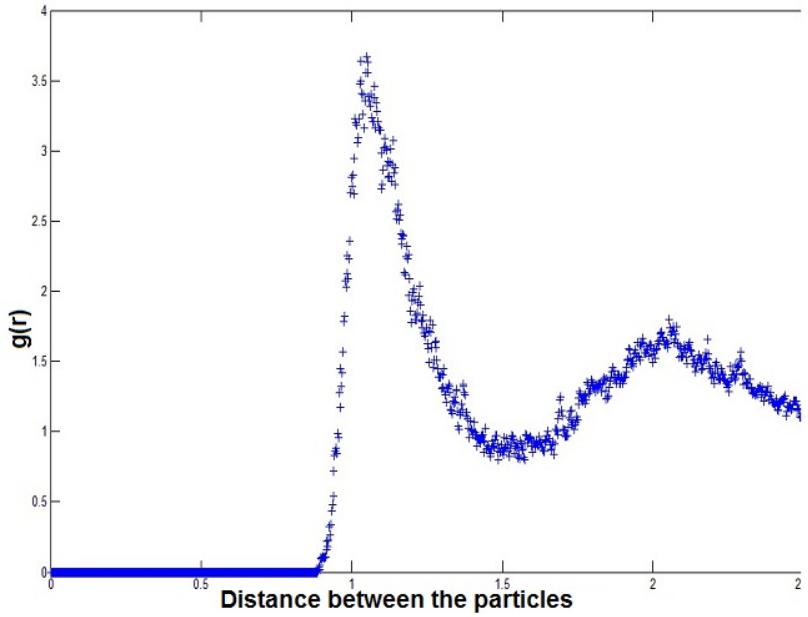

Figure 1.3: Radial distribution function of distance between the particles

and for velocity updates subtracting Eq. 1.20 and Eq. 1.21

$$r(t + \Delta t) - r(t - \Delta t) = 2v(t)(\Delta t) + O((\Delta)^3) \tag{1.23}$$

hence ignoring the higher order terms $v(t) = (r(t + \Delta t) - r(t - \Delta t))/(2\Delta t))$.

1.4.1 Results

The kinetic energy increases first and then fluctuates about an equilibrium value. The temperature calculated from the kinetic energy is $T = 1.5 \pm .05$ (Fig 1.2). The potential energy falls and then saturates. The total energy is constant as expected. We also calculate the radial distribution function which gives information about structural properties (Fig 1.3).

Chapter 2

Active Brownian particles

2.1 Individual Dynamics

We now introduce "activity" in the dynamics of the passive Brownian particles. Before looking at a collection of active particles, we first investigate how activity can affect the dynamics of a single particle. As we have mentioned earlier an active particle absorbs energy from environment and converts it to kinetic energy in the form of movement. There are several ways activity can be introduced into passive Brownian motion. Here we discuss one such model which is called the Depot model.

2.1.1 Depot model

The key features of this model are:

1. The particle take up the energy from the environment where $q(x)$ is space dependent flux of energy into the depot.

2. Particle dissipates energy which is assumed to be proportional to internal energy rate of this dissipation, c, is assumed to be constant.

3. Conversion rate of internal energy into kinetic energy is function of velocity, $d(v)$.

Hence, change of internal energy, e, can be written as:

$$\frac{de(t)}{dt} = q(x) - ce(t) - d(v)e(t) \tag{2.1}$$

Example We can make the assumption that

$$d(v) = d_2 v^2, \quad d_2 > 0 \tag{2.2}$$

The system has total energy of E_t as sum of internal energy $e(t)$ and mechanical energy $E_0(t)$, where $E_0(t)$ can be written as

$$E_0(t) = \frac{mv^2}{2} + U(r) \tag{2.3}$$

Now, change in the mechanical energy can be found by increase in kinetic energy due to conversion of internal energy into it and decrease in mechanical energy due to dissipation in the environment

$$\frac{dE}{dt} = [d_2 e(t) - \gamma_0]v^2 \tag{2.4}$$

γ_0 is the dissipative frictional force coefficient and second term of left hand side is written using power=force×velocity, frictional force is velocity dependent force with coefficient γ. Now, using Eq. 2.3 we can write

$$\frac{dE}{dt} = mv\dot{v} + \nabla U\dot{r} \tag{2.5}$$

and hence

$$mv\dot{v} + \nabla U v = [d_2 e(t) - \gamma_0]v^2 \tag{2.6}$$

factoring the common v out and adding the random force with Gaussian white noise η

$$m\dot{v} + \nabla U \pm \gamma_0 v = d_2 e(t)v + m\sqrt{2D}\eta \tag{2.7}$$

which is the Langevin equation for this system. Now we can see by comparing Eq. 2.7 and Eq. 1.2 that a new term $d_2 e(t)v$, has been added in the active case. This is like an acceleration in the direction of motion. This term comes about from the conversion of internal energy into kinetic energy. The system is driven into non-equilibrium in addition to the stochastic term.

2.2 Collective motion

We will now try to model the collective dynamics of active Brownian Particles. To do this we need to understand the nature of interaction of the particles. There could be "polar" interaction where the particles collectively move in the specific direction - examples are birds, schools of fishes. There could be apolar interactions such as self propelled rods which try to align along their long axis but can move parallel or anti parallel directions. Self propelled rods are examples of the polar particles with polar interactions. In this chapter, we are going to discuss about the Viscek model which looks at polar particles with polar interactions.

2.2.1 Viscek model

The main points incorporated by Vicsek are [9]:

1. Initially, there are large number of point particles having the same velocity v_0 and random directions θ.

2. The particle moves in a direction which is dependent on the direction of its neighbors. However the extent to which a particle feel it's neighbors is really small (R_0) as compared to total size of the flock (L). Therefore the interaction are purely short ranged.

3. The following is not prefect, due to collisions or other errors, there is a noise in the system.

4. The model has complete rotational symmetry: the flock is equally likely, a priori, to move in any direction.

The simulation steps [10] thus could be chosen according to definition of model. The i^{th} particle is situated at position $r(t)$ in a two-dimensional plane, at integer time t. These particles move in the direction which is average of the direction of all the particles within the radius of R_0, including the particle itself, on the previous time step (Eq. 2.8,Eq. 2.9). The distance R_0 is assumed to be appropriately smaller than L, the size of the flock. The direction the particle actually moves on the next time step differs from the above described direction by a random angle $\eta(t)$. This random angle is chosen with a uniform probability from the interval $[-\eta/2, \eta/2]$. Each particle then, on the next time step, moves in that direction a distance $v_0\Delta t$, where the speed v_0 is the same for all particles.

$$\theta_i(t + \Delta t) = \langle\theta_j(t)\rangle_n + \eta_i(t), \tag{2.8}$$

$$r_i(t + \Delta t) = r_i(t) + v_0(cos(\theta(t + \Delta t)), sin(\theta(t + \Delta t)))\langle\eta_i(t)\rangle \tag{2.9}$$

the $\langle\rangle_n$ symbolizes the average over the neighbors and $\theta_i(t)$ is the angle of the direction of motion of the i^{th} particle on the time step that ends at t.

2.2.2 Results

The images show how the system is initiated homogeneously with random direction and after some time the system starts moving in a direction breaking rotational symmetry and spatial symmetry (Fig2.1). The packing fraction defined by

$$\rho = \frac{N\pi r_0}{L^2} \tag{2.10}$$

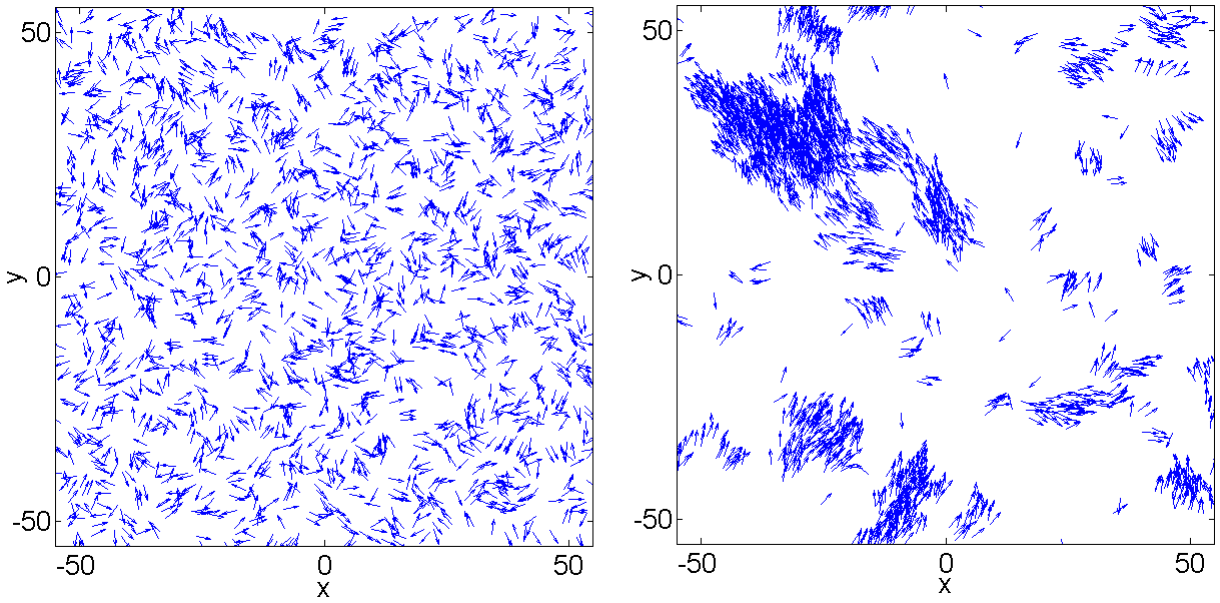

Figure 2.1: Vicsek model $\rho = 2$,$v_0 = 1$(a)Initial state at $t = 0$(b)State after $t = 10^4$

This quantity gives the information about how closely the particles are packed and hence gives the quantitative idea of amount of particles feeling each other at certain point in time. The images are shown for a quite low ρ, we can see a big cluster on the top left of the (Fig 2.1 b). This cluster decides the main polarization of the system at the moment. However there are small cluster moving in different directions which are decreasing the net polarization of the system. These small cluster can almost always be seen at low $\rho's$, but at higher a ρ these small cluster are also caught up by the big cluster and the net polarization is quite high. The phases can be characterized by the mean velocity defined by

$$v_{mean} = \frac{1}{N}\sum_{i=1}^{N}\theta_i \tag{2.11}$$

This quantity acts as order parameter. It's high in a ordered state and almost negligible in the disordered state. The phase diagram (Fig 2.2) of the system under the control parameter of noise intensity η shows the transitions happening at around $\eta = 0.62$. The diagram also reveals the fact that at low $\rho's$ the polarization is low, this can be realized as the mean velocity is decreasing as we decrease the ρ.

Thus in the Vicsek model, for small ρ and η, the particles move in small groups. At higher densities but for small noise, the motion becomes ordered and all particles move in the same direction. This direction is the one that is chosen spontaneously.

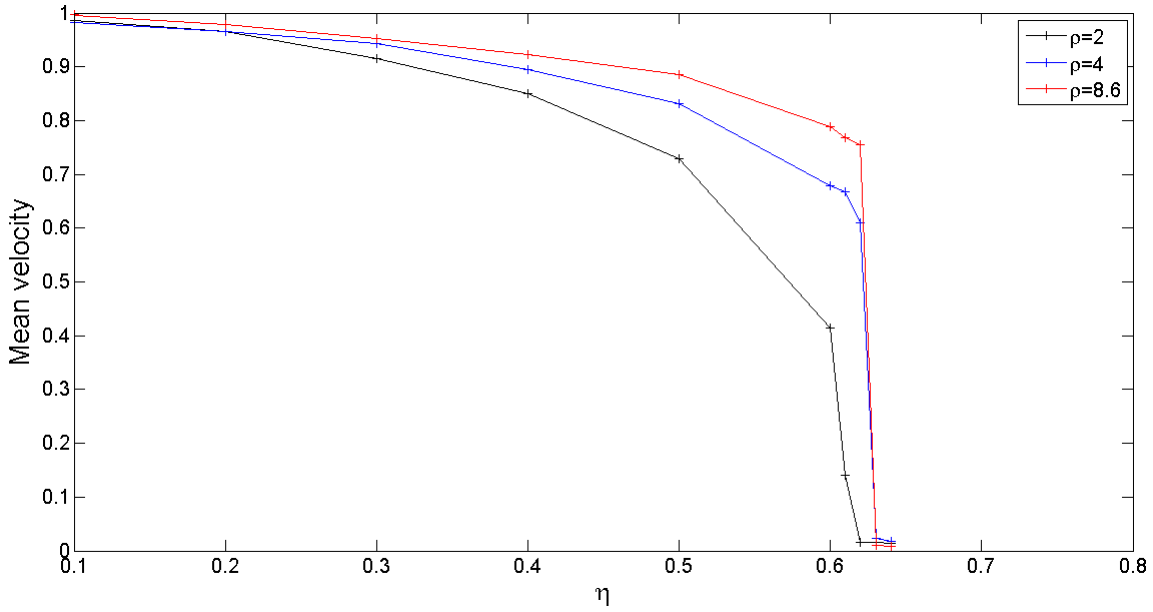

Figure 2.2: Mean velocity Vs η for constant speed Vicsek model

2.3 Variable Speed Model

In the above system the speed for every particle kept constant. It's been observed that in experiments (Fig 2.3), that the speed of the particles vary from particle to particle, depending on it's neighborhood polarization [12]. Therefore we define a quantity χ called as local polarization.

$$\chi = \frac{1}{N_j} \left| \sum_{j}^{N} v_i \right| \tag{2.12}$$

j are particles within radius r_0 around the particle.This quantity can have maximum value of 1 in the case when all the particles in the neighborhood (at a distance less than r_0) are moving in the same direction or would be close to 0 if all the particles in the neighborhood have random orientations.

The system has N polar particles started from random positions and random orientations in 2-dimensions moving with direction rule given by

$$v_i(t) = v_M \left(\chi_i(t)\right)^{\gamma} \tag{2.13}$$

where v_M is maximum speed that a particle may reach. If the particles within the interaction radius are moving in the same direction, them χ is close to 1 and the particle move with speed v_M. However, if the particle directions are completely random, then $\chi \approx 0$ and the particles become essentially immobile. The particles which are isolated (away from the clusters) move with the maximum speed as there is no particle in the vicinity to damp it's

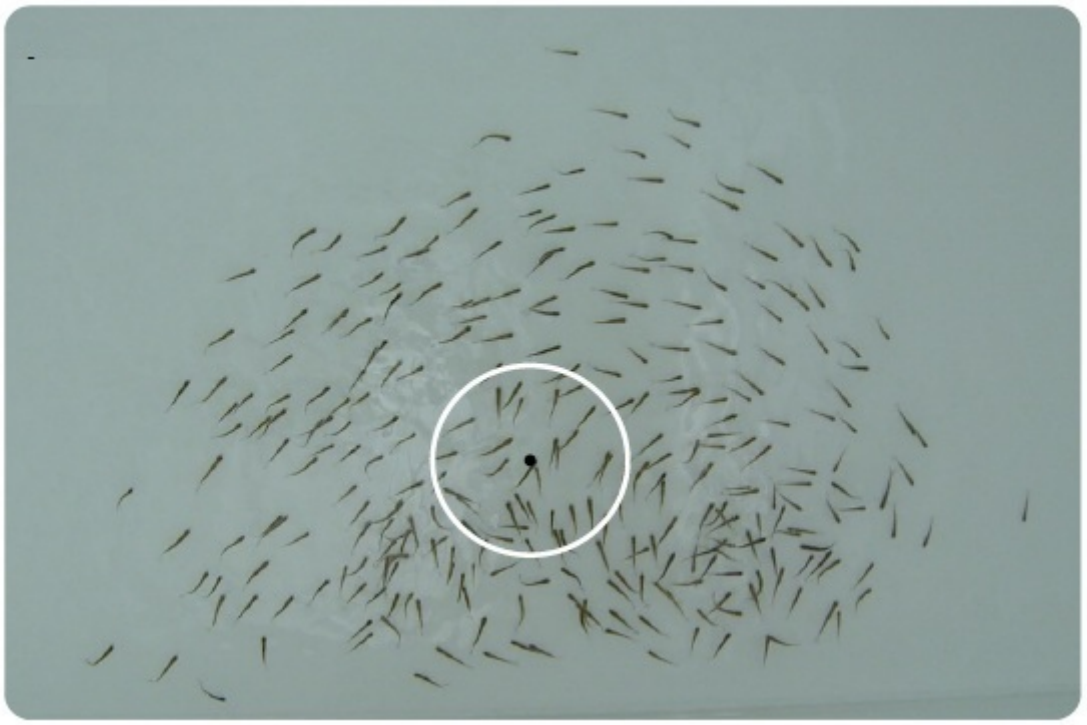

Figure 2.3: To see if the above model is sufficient to mimic the real system a group of gold shiner fishes was used by([13]). Automated video tracking was used to track the movement of fishes. The white circle shown in the figure shows the chosen interaction radius to find out the χ.(reproduced from [12])

velocity with.

2.3.1 Simulations

N particles are simulated over 2 dimensional space with periodic boundaries, positions are updated with the rule

$$\vec{r_i}(t+\Delta) = \vec{r_i}(t) + v_i(t)\hat{\theta_i}(t) \tag{2.14}$$

where v_i is self propulsion velocity of the i^{th} particle at time t and $\hat{\theta_i}$ is the direction of this self propulsion velocity.Time step Δt is taken to be 1. The direction update is given using the rule

$$\hat{\theta_i}(t+\Delta) = \frac{1}{W_i}\left(\sum_j \hat{\theta_j}(t) + N_i\eta_i\right) \tag{2.15}$$

where W_i is normalization constant, which keeps the quantity inside the bracket as unit vector or direction,j are similar neighborhood particles η_i here are randomly oriented vectors with $\vec{\theta_i} = \eta(cos(\theta)\hat{x} + sin(\theta)\hat{y}$ and hence θ is uniformly distributed random variable in the interval $[-\pi, \pi]$ Velocity of particles actually depend on the $\gamma/2$ power of the number of

particles in the neighborhood of the particle This can be easily proven by the analyzing relation Eq. 2.13.

$$v_i(t) = v_M[\left(\frac{1}{N_j}|\sum_j \hat{\theta}_j(t)|\right)^{\gamma} \tag{2.16}$$

Expanding the modulus terms

$$|\sum_j \hat{\theta}_j(t)| = \sqrt{N_j(\hat{\theta}_j.\hat{\theta}_j) + \sum_{j\neq j'} \hat{\theta}_j.\hat{\theta}'_j} \tag{2.17}$$

as $\theta'_j s$ are unit vectors hence $\hat{\theta}_j.\hat{\theta}_j$ is equal to 1. Therefore

$$v_i(t) = v_M \left(\frac{1}{N_j^{\gamma/2}}[1 + \frac{1}{N_j}\sum_{j\neq j'} \hat{\theta}_j.\hat{\theta}'_j]\right)^{\gamma/2} \tag{2.18}$$

now the term in $\sum_{j\neq j'} \hat{\theta}_j.\hat{\theta}'_j$ is very small as compared to N_j.So to the first order the relations we obtain

$$v_i(t) = \frac{v_M}{N_j^{\gamma/2}} \tag{2.19}$$

Hence the more the neighbors less is the velocity of the particle which has been observed in the results.

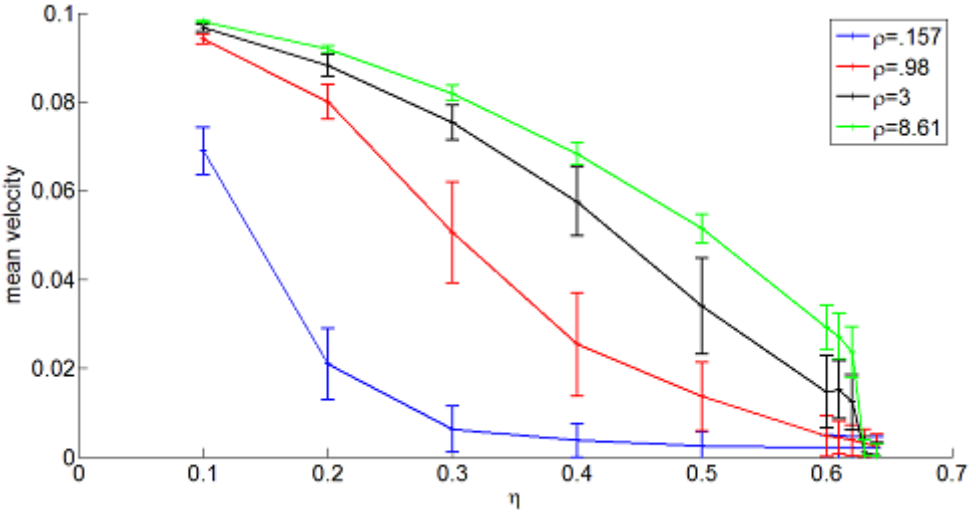

Figure 2.4: Mean velocity Vs η at different $\rho' s$.

2.3.2 Results

Here we choose, $N = 2000$,$\gamma = 6$. The system transits from the ordered state (Fig 2.6) to static state (Fig 2.8) when noise intensity η is increased. When at low noise intensity

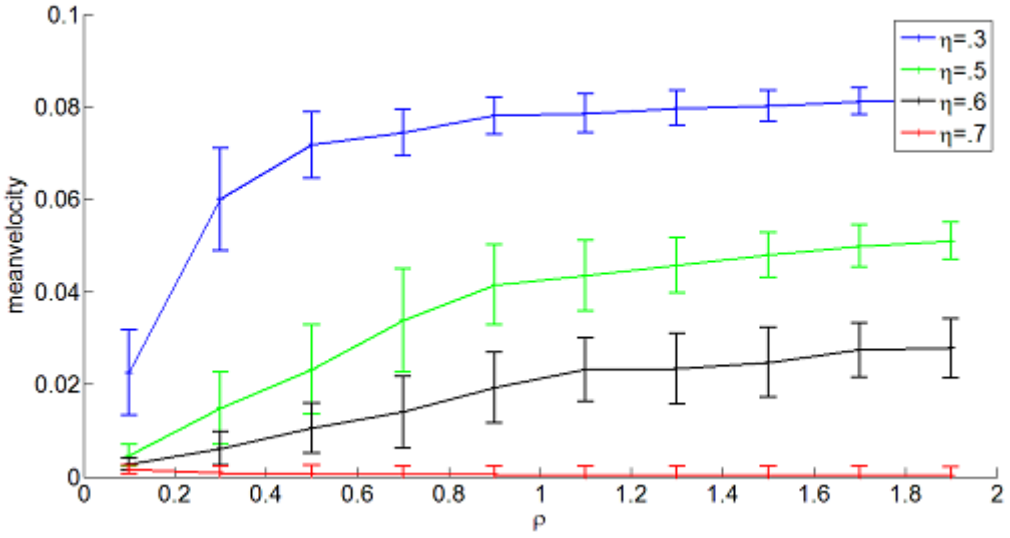

Figure 2.5: Mean velocity Vs ρ at different $\eta's$.

the particles start moving in same direction as noise is not enough to compete with the aligning interaction but as the noise increases the velocity of ordered state starts decreasing and finally at high noises the particles become almost immobile as now aligning interaction are negligible in comparison to the noise intensity. An interesting phenomenon occurs at intermediate noise (Fig 2.7) when the transition just starts taking place, the system shows bistable solution in the distribution of velocity. There are two types of particles that are present in the system one with low velocity which lie in the static clusters and one with the high velocity which are away from clusters. This situation can be seen at a vary specific value of noise. For parameter that we chose, this happens at $\eta = 0.625$. At 0.63 the system is in disordered state and at 0.62, it's still a ordered state. Finally we use mean velocity (Eq. 2.11), to show that it act as a order parameter and characterize the transition (Fig 2.5) and (Fig 2.4), as it reduces to low values when the system is static. The mean velocity can be seen decreasing as the noise intensity is increased the transitions occurs at around $\eta = .62$ similar to constant velocity Vicsek model. Hence transition point is same for both variable speed and constant speed model (Fig 2.2, Fig 2.4). However it can be observed that the errors increase as the noise intensity which is evident from the fact that the noise is always trying to increase the disorderness of the system. The maximum mean velocity which is reached when noise is low can be seen to be decreasing with the decrease in the ρ, this is because of same fact explained in the previous section. The system forms smaller groups moving in random direction decreasing the mean velocity of the system (Fig 2.5). The graphs of mean velocity vs ρ show that higher density is required for the higher mean velocity or a proper ordered state.It also shows that change of mean velocity per unit increase in ρ higher for lower noises(slope of curve initially),therefore it would require higher densities to reach

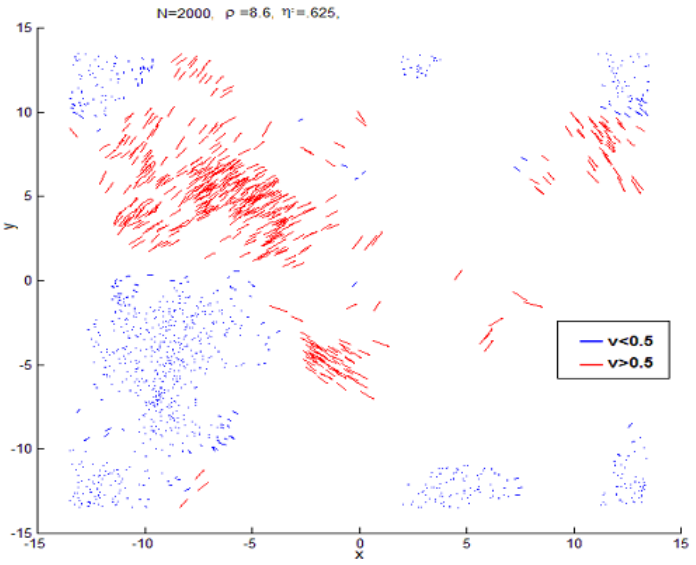

Figure 2.6: After 300000 timesteps at noise intensity 0.1.

a ordered in the case of high noises.

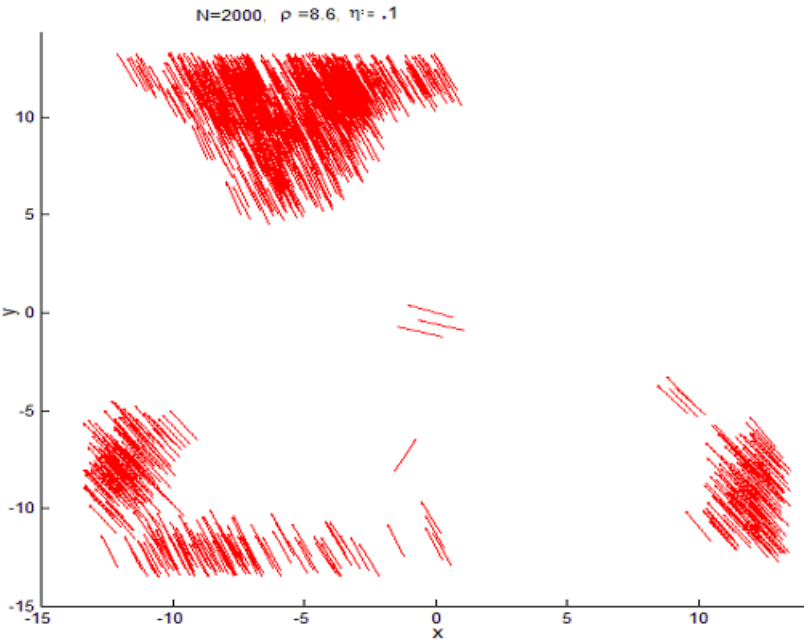

Figure 2.7: After 300000 timesteps at noise intensity 0.625.

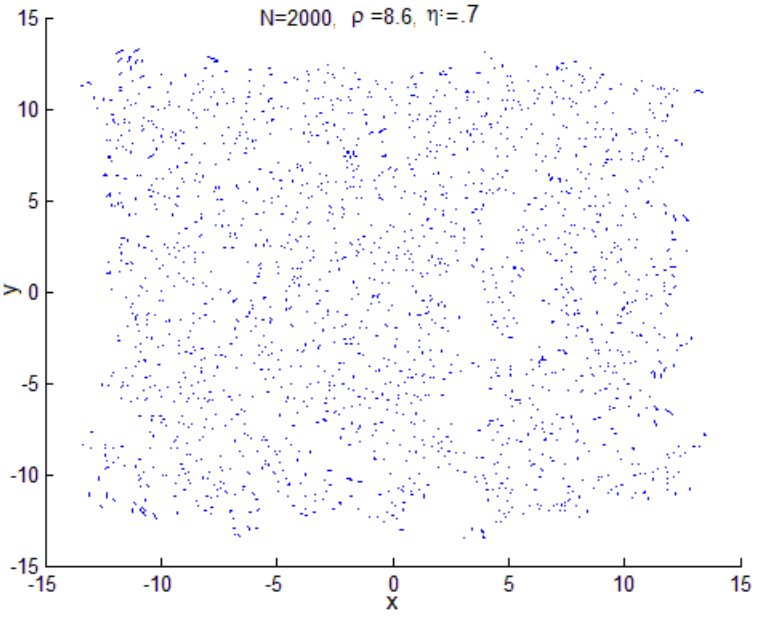

Figure 2.8: After 300000 timesteps at noise intensity 0.7.

Chapter 3

Active Driven Disks

We now consider another class of active Brownian system where there is interaction is purely repulsive and without any aligning interactions.Thus the key features of the model are

1. The particles are in form of disks moving in 2 dimensions with repulsive potential.
2. The direction of self propulsion force is same for all the particles is not affected by the direction of neighboring particles.
3. There is rotational noise in the system but translation noise is ignored,
4. The system is overdamped

Force due to repulsion is modeled by[14]

$$F_{ij} = k(2a - r_{ij})\vec{r}_{ij} \quad if \quad r_{ij} < 2a \tag{3.1}$$

$$= 0, \quad otherwise$$

where r_{ij} is the distance between particles i and j and $\vec{r_{ij}}$ is position vector separating the particles and a is the radius of disk.
and Langevin equation can be written as

$$\frac{\partial r_i}{\partial t} = v_0\hat{v}_i + \mu \sum_{j \neq i} F_{ij} + \eta_i^T(t) \tag{3.2}$$

$$\frac{\partial \theta_i}{\partial t} = \eta_i^R(t) \tag{3.3}$$

where r_i is the position, θ_i is the propulsion direction ,$\hat{v}_i$ is representing the unit vector in self propulsion direction $\hat{v}_i = (cos(\theta_i), sin(\theta_i))$, v_0 is the self propulsion speed, $\eta_i's$ are Gaussian white noise Eq. 1.1, and μ is the mobility.

3.1 Simulations

First for simulating these systems we needed to non dimensionalize the equations Eq. 3.2 and Eq. 3.3. First we need to look at the delta relation of noise.

$$\langle \eta_i^{\alpha}(t)\eta_j^{\beta}(t')\rangle = 2D\delta_{ij}\delta_{\alpha\beta}\delta(t-t') \tag{3.4}$$

here $\delta(t-t')$ has a dimension of t^{-1} and $D = \mu k_B T$, therefore, it's obvious to scale time, t, as $1/\mu k$ and length can be scaled by the particle radius a. Hence keeping $a = 1$, and $\mu k = 1$; we can write non dimensionalized equations (without translational noise) in components as to be used in the simulations.

$$\frac{\partial \tilde{x}_i}{\partial \tilde{t}} = \left[\tilde{v}_0 cos(\theta_i) + \sum_{j\neq i}\left(\frac{2}{r_{ij}} - 1\right)x_{ij}\right] \tag{3.5}$$

$$\frac{\partial \tilde{y}_i}{\partial \tilde{t}} = \left[\tilde{v}_0 cos(\theta_i) + \sum_{j\neq i}\left(\frac{2}{r_{ij}} - 1\right)y_{ij}\right] \tag{3.6}$$

$$\frac{\partial \tilde{\theta}}{\partial \tilde{t}} = \sqrt{2D}\eta \tag{3.7}$$

where $\tilde{x}_{ij}^2 + \tilde{y}_{ij}^2 = \tilde{r}_{ij}^2$,$\tilde{r}_{ij} = r_{ij}/a$ and $\tilde{t} = t/\mu k$.

Cell list algorithm

As the system can consists of many particles(upto 10^4), and due to particle-particle interactions, this will require N^2 operation in calculating forces which is really time expensive. Therefore, we needed to apply cell list algorithm. In this algorithm we divide the simulation space into cells with an edge length greater than or equal to the cut-off radius of the interaction between particles. The particles are then distributed in the particular cells. When we choose a particle and find its interaction with neighbors, then it is calculated only with particles in the same cell and with particles in the nearest neighboring cells (Fig 3.1).

3.1.1 Results

System is simulated on 2-dimensional space with periodic boundaries on densities defined by $\rho = N\pi a^2/L^2$, where L is the length of box of simulation box. Simulations are performed for different values of propulsion velocities and different densities. This system is seen to be showing two different phases one is the clustered(solid-like) and one with the gas-like at density higher than a critical density ρ_c and above critical propulsion velocity (Fig 3.2 , Fig 3.3). Once the cluster is formed it keeps moving by keeping the size almost constant. However,the cluster don't move as a whole but some particles on the surface

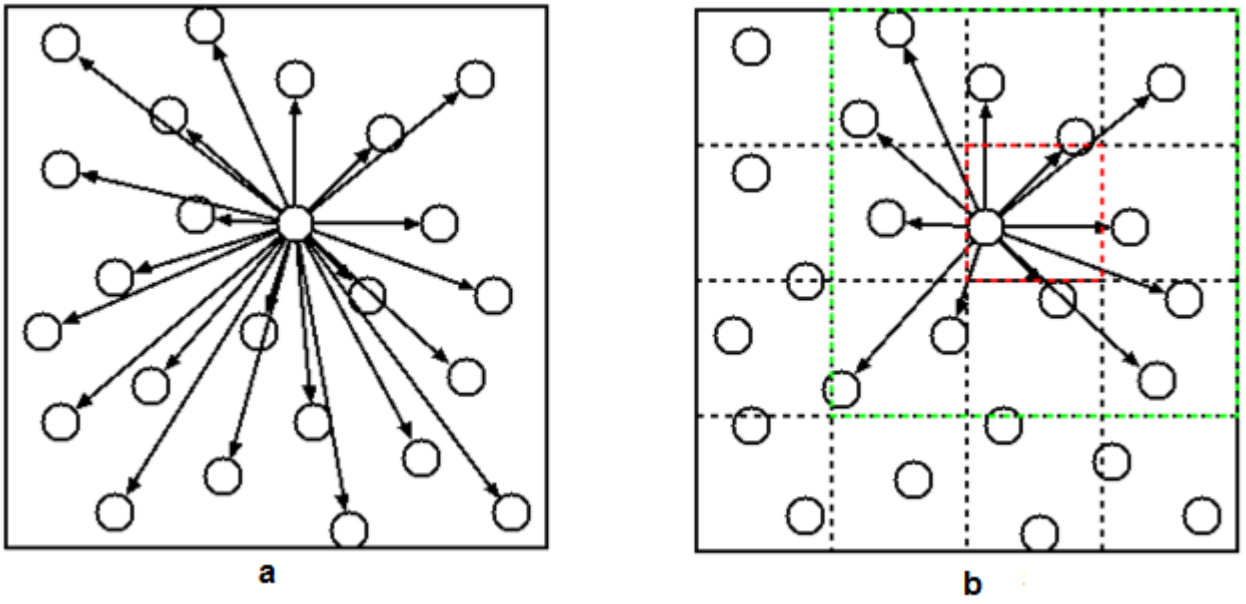

Figure 3.1: (a) Comparisons without cell list (b) Comparisons with cell list (Reproduced from Wikipedia)

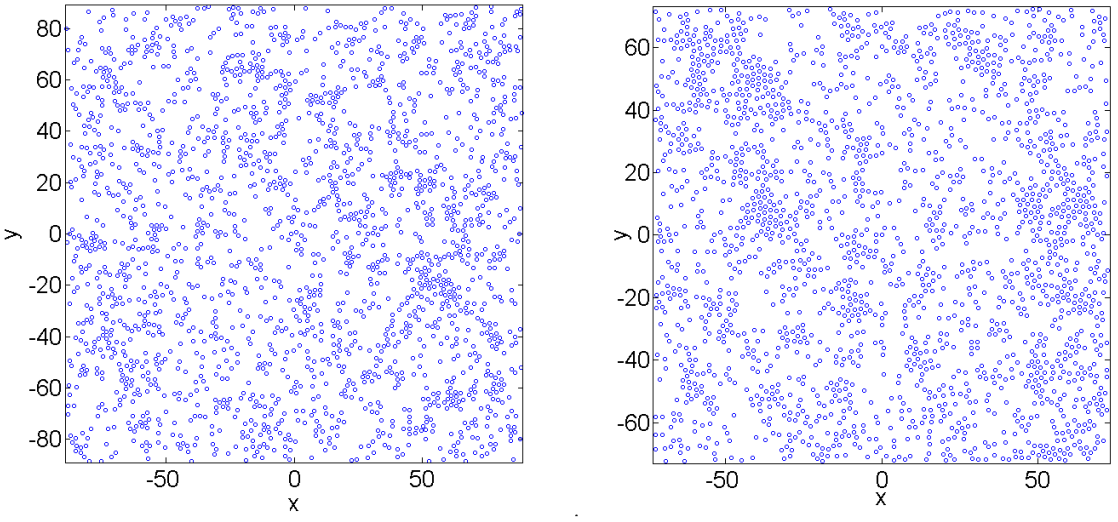

Figure 3.2: State after 10^8 timesteps (a) for $\rho = 0.2$, $S = .0005$ (b) for $\rho = .3$, $S = .0012$.

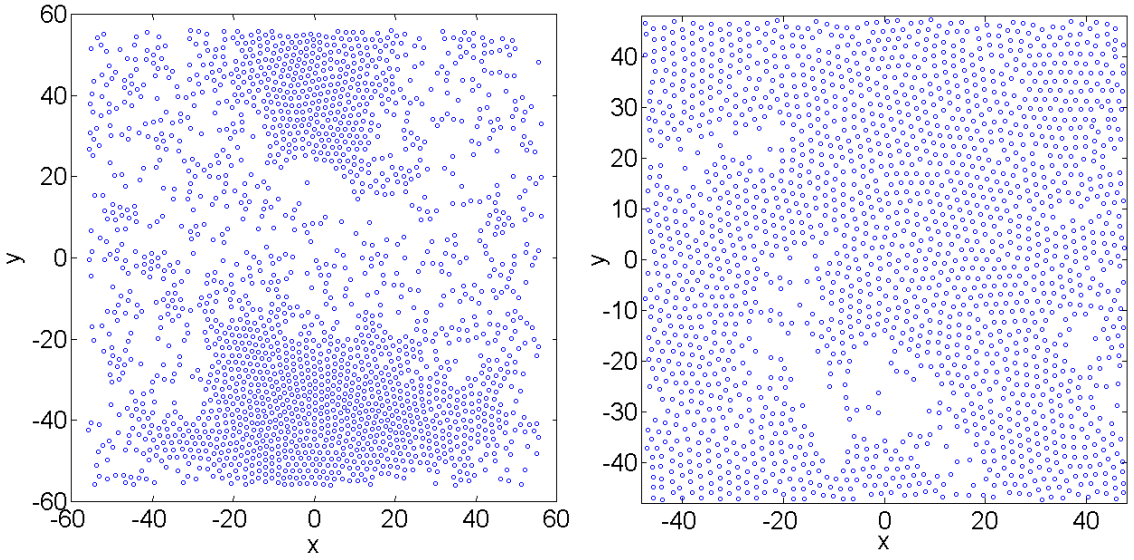

Figure 3.3: State after 10^8 timesteps (a) for $\rho = 0.5$, $S = .53$ (b) for $\rho = .7$, $S = .85$.

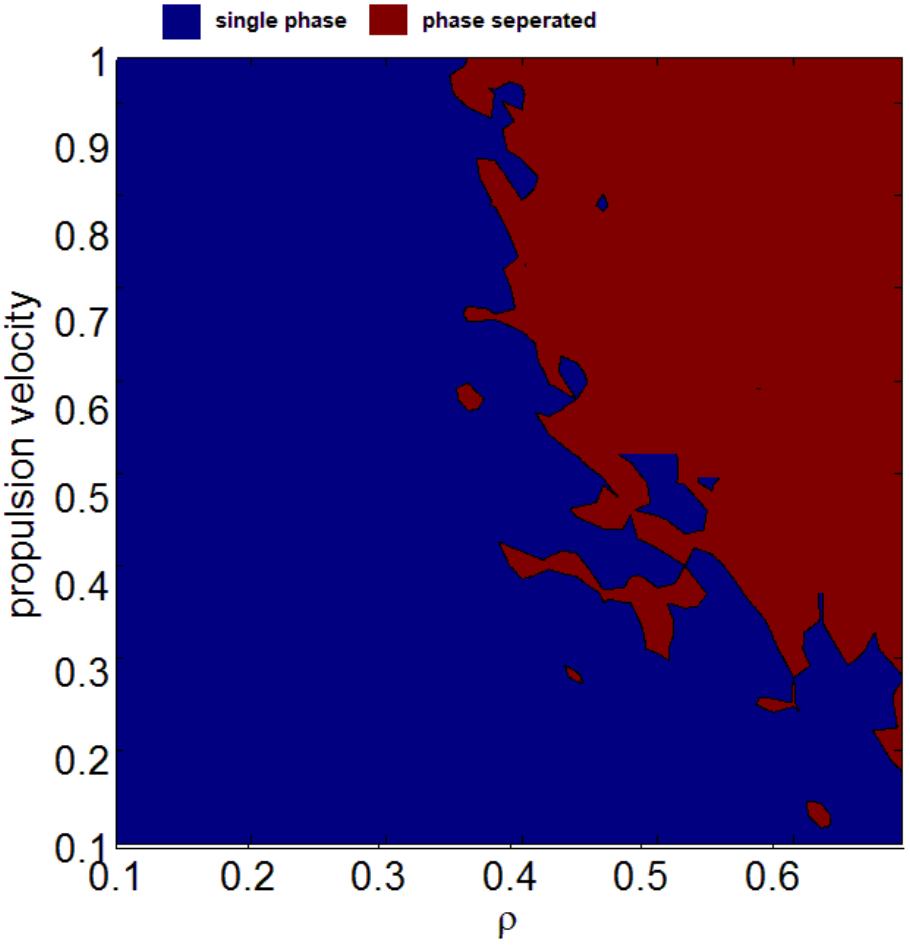

Figure 3.4: Phase diagram for the active driven disks

constantly leave the cluster and almost equal number of them are constantly added. Hence size is kept constant. However, the direction from which they add and leave keep changing, as a results the cluster moves all over the space. The phase diagram is explored using the quantity S [16] which is fraction of particles which are at a distance $\leq 2a$, averaged over large timesteps. This value is calculated at discrete points on the propulsion velocity vs density space, then linearly interploted to calculate the phase diagram shown in (Fig 3.4). S can be seen to be appropriately characterizing the clustered state in these images.

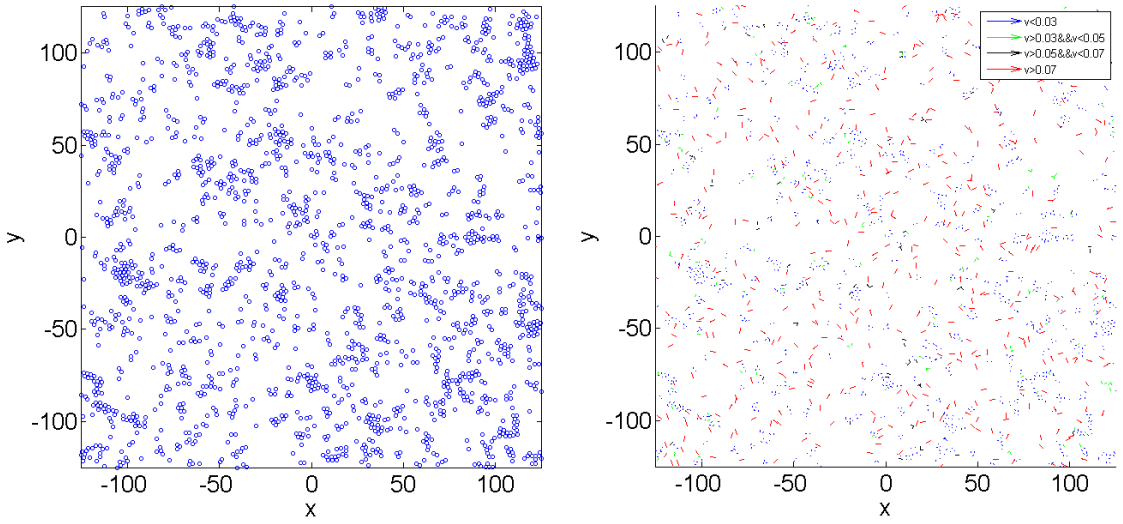

Figure 3.5: State after 10^7 timesteps for $\rho = 0.1$ (a) Spatial distribution (b) Corresponding spatial distribution of velocity.

3.2 Variable Speed

Similar to Viscek model, these disks can also be modeled with the variable velocity and mean velocities can be calculated to see if transitions occur.

3.2.1 Results

The system is started with random velocities and the random positions on a periodic boundaries. Similar cell listing algorithm is used for θ averaging, to calculate the velocity of particle in some point in space. It is seen that particle still cluster spatially but the gas phase is less well defined in variable speed regime. Interestingly, for the variable speed model, the critical density where phase separation happens is lower than that in the constant speed model. By comparing the (Fig 3.2 b) and (Fig 3.6 a) we can clearly see that the cluster formation in the case of variable speed is quite high compared to formation at constant speed.

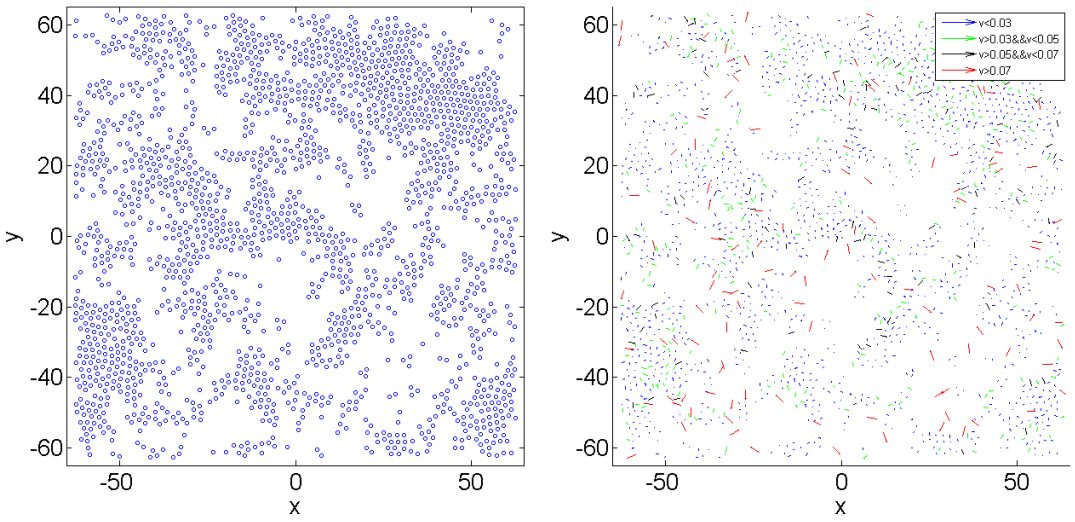

Figure 3.6: State after 10^7 timesteps for $\rho = 0.3$ (a) Spatial distribution (b) Corresponding spatial distribution of velocity.

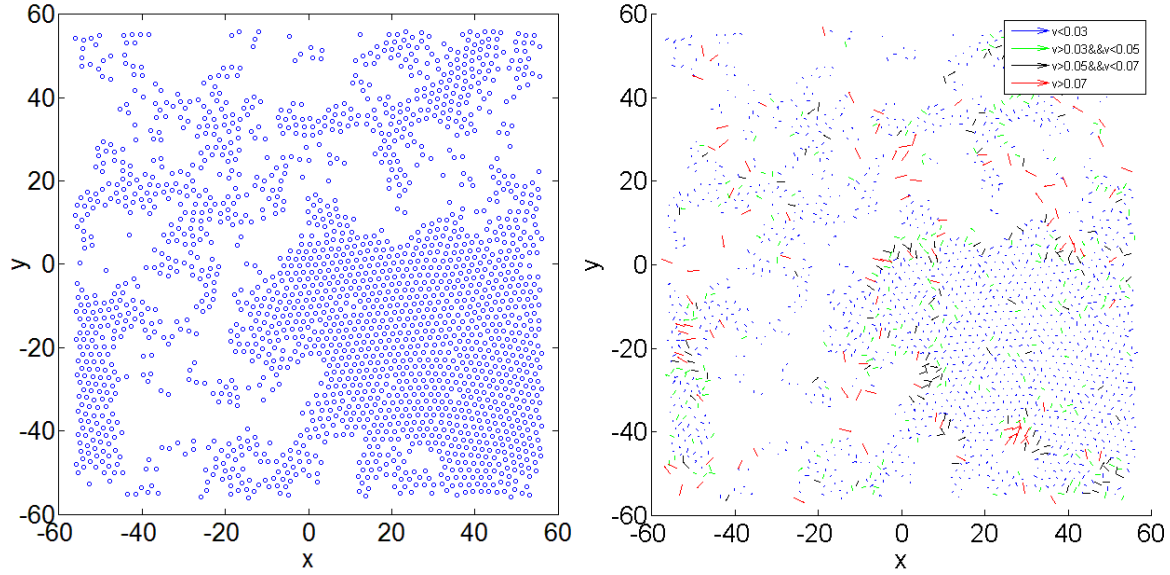

Figure 3.7: State after 10^7 timesteps for $\rho = 0.4$ (a) Spatial distribution (b) Corresponding spatial distribution of velocity.

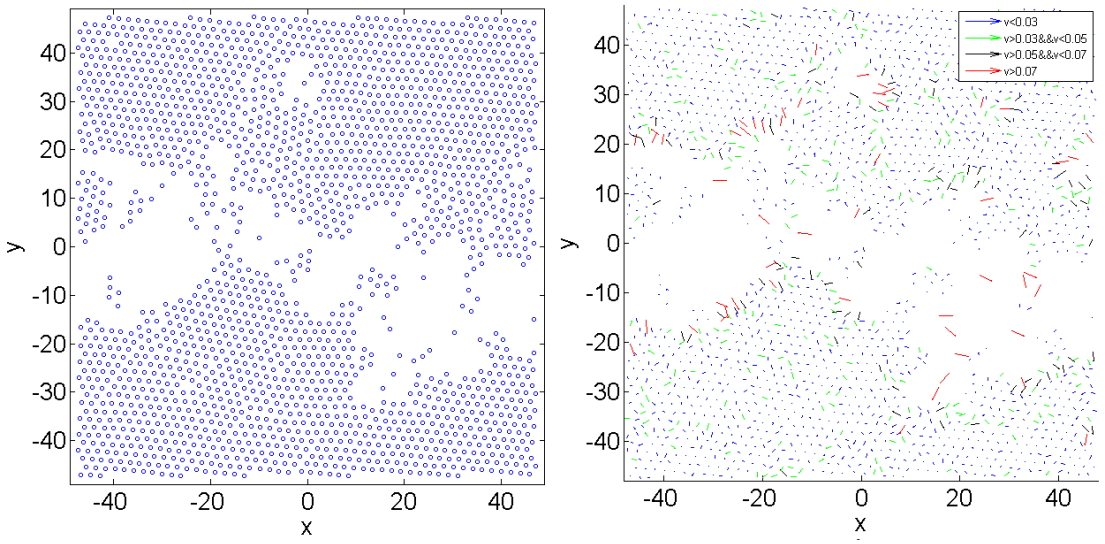

Figure 3.8: State after 10^7 timesteps for ρ=0.6 (a) Spatial distribution (b) Corresponding spatial distribution of velocity

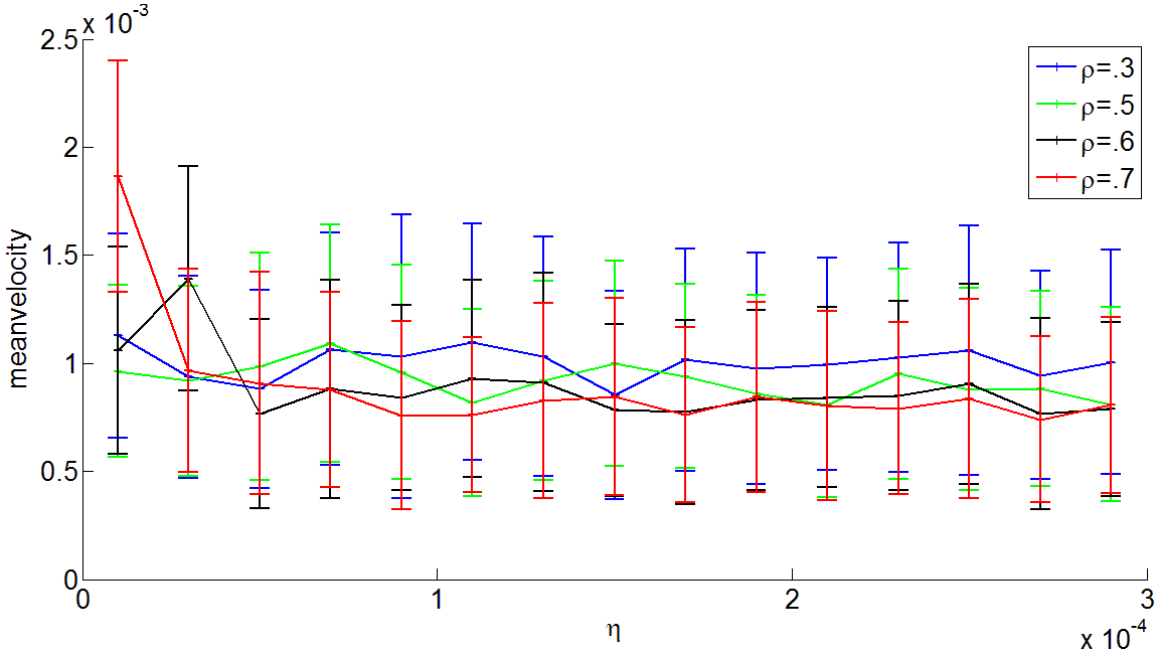

Figure 3.9: Mean velocity vs η for different ρ values.

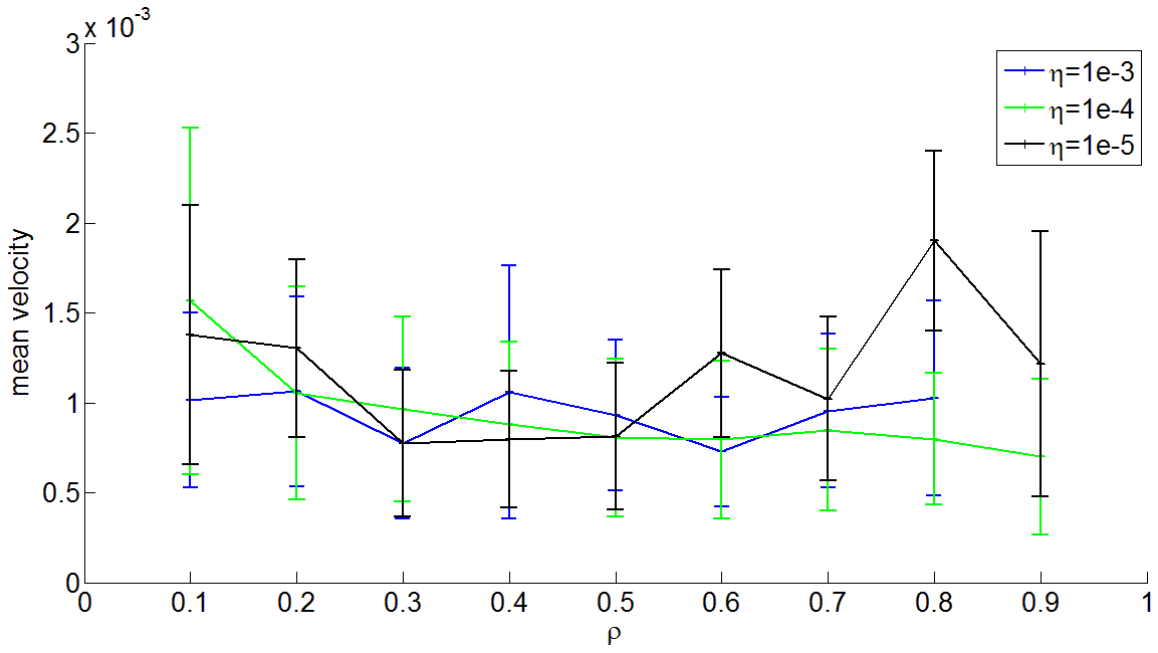

Figure 3.10: Mean velocity Vs ρ for different η values.

Moreover the the particles at the surfaces of the cluster can be seen having high velocity in the direction opposite to the cluster and there is very low density around the surface of the cluster. The reason can attributed to these outgoing particles with high velocity as they escape the cluster. The spatial distribution of velocity shows that particle in the clusters have a very low velocity and they are mostly immobile; the particles which are away or isolated are moving with high almost at maximum velocity(Fig 3.5 , Fig 3.6 , Fig 3.7 , Fig 3.8). The mean velocity vs density curves and mean velocity vs noise curves show almost nothing or no pattern to suggest any transition (Fig 3.10 , Fig 3.9). This is expected because the particles here are disks without any aligning interactions. The graphs are tried to various ranges of noises to look for the pattern, but mean velocity keeps fluctuating with high errors(standard deviation). The reason may be attributed to the missing aligning interactions.

Appendix A

Program codes

Preliminary subroutines required for implementing the cell list algorithm

1. **Map subroutine**: Decides the 8 neighbors(or 4 in the case of active driven disks) of a cell to for the comparisons required for interaction calculations.

```
void mapev(void){
for(i=1;i<nc1;i++){      \\nc is the number of the cells
  for(j=1;j<nc1;j++){   \\per side of simulation box
                int imap=((i-1+nc)%(nc)+((j-1+nc)%nc)*nc)*8;
                mapv[imap+1]= 1+(i+nc)%(nc)+((j-1+nc)%nc)*nc;
                mapv[imap+2]= 1+(i+nc)%(nc)+((j+nc)%nc)*nc;
                mapv[imap+3]= 1+(i-1+nc)%(nc)+((j+nc)%nc)*nc;
                mapv[imap+4]= 1+(i-2+nc)%(nc)+((j+nc)%nc)*nc;
                mapv[imap+5]= 1+(i+nc)%(nc) + ((j-2+nc)%nc)*nc;
                mapv[imap+6]= 1+(i-2+nc)%(nc) + ((j+nc-1)%nc)*nc;
                mapv[imap+7]= 1+(i-2+nc)%(nc) + ((j-2+nc)%nc)*nc;
                mapv[imap+8]= 1+(i-1+nc)%(nc) + ((j-2+nc)%nc)*nc;
                }
        }
}
```

2. **Link subroutine**: Required for assigning the cells to the respective particles every time after coordinates of the cells are changed

```
void link(void){
for(i=1;i<nc21;i++)   \\nc21 is the total number
  head[i]=0;           \\of cells present
for(i=1;i<N1;i++){
 int icell=1+int((cord[0][i]+(box1/2))*(double(nc)/box1))..
        +int((cord[1][i]+(box1/2))*(double(nc)/box1))*nc;
        list[i]= head[icell];
        head[icell]= i;
        }
}
```

3. **Average subroutine**: This routine calculates the aligning interactions and local polarization χ within specified radius

```
void average(void){
        for(i=1;i<nc21;i++){
                int I= head[i];
                //loop over all the molecules in the cell
                        while(I>0){
                        for(di=0;di<2;di++)
                         cordH[di]=cord[di][I];
                         thetafc=0;
                         thetafs=0;
                         counti=0;
//for particles in the same cell
                 int J= head[i];
                     while(J>0){
                         double r2s=0;
                         if(J<N1){
                         for(di=0;di<2;di++){
                         xd[di]=cordH[di]-cord[di][J];
                     xd[di]=xd[di]-box*round(double(xd[di]/box));
                         r2=pow(xd[di],2);
                         r2s=r2s+r2;
                }
                        if(r2s<rC2){
                        thetafc += (thetao[0][J]);      //thetao are the propulsion
                        thetafs += (thetao[1][J]);    // direction from a timestep back
                        counti++;
                                }
                        }
                        J= list[J];
 //for the particles in the neighboring cells
     int jcell0=8*(i-1);
     //looking for all the neighboring cells with present head of cell chosen
           for(int nabor=1;nabor<9;nabor++){
                 int jcell= mapv[jcell0 + nabor];
                  J = head[jcell];
                        while(J!=0){
                        double r2s=0;
                        if(J<N1){
                        for(di=0;di<2;di++){
                        xd[di]=cordH[di]-cord[di][J];
                   xd[di]=xd[di]-box*round(double(xd[di]/box));
                         r2=pow(xd[di],2);
                         r2s=r2s+r2;
                                }
                   if(r2s<rC2){
                         thetafc += (thetao[0][J]);
```

```
                        thetafs += (thetao[1][J]);
                        counti++;
                                        }
                        }
                    J=list[J];
                                    }
                          }
tot[I]= (double)sqrt(double((thetafs*thetafs)+(thetafc*thetafc)));
double f =  3.142*(2*(rand()/(double)RAND_MAX)-1);
thetafc += nL*counti*cos(f);
thetafs += nL*counti*sin(f);
double  toti=(double)sqrt(double((thetafs*thetafs)+(thetafc*thetafc)));
thetafs= thetafs/toti;
thetafc= thetafc/toti;
tot[I]=tot[I]/counti;
tot[I]=pow(tot[I],gamma);
theta[0][I] = thetafc;
theta[1][I] = thetafs;

                    }
    I= list[I];

        }
        for(i=1;i<N1;i++){
          thetao[0][i]=theta[0][i];
          thetao[1][i]=theta[1][i];}
}
```

4. **Force subroutine** This routine calculates the potential generated force between the particles in the active disks model.

```
void force(void){
                for(k=1;k<N1;k++){
                for(di=0;di<2;di++){f[di][k]=0;}}  //zeroing the force
                for(i=1;i<nc21;i++){
                  int I= head[i];
                //loop over all the molecules in the cell
                  while(I>0){
                  for(di=0;di<2;di++){
                        cordH[di]=cord[di][I];
                        fH[di]=f[di][I];
                        }
                    int J = list[I];
                    while(J>0){         //for particles in the same cell
                          for(di=0;di<2;di++){
                          xd[di]=cordH[di]-cord[di][J];
                          xd[di]=xd[di]-box*round(double(xd[di]/box));
                          r2=pow(xd[di],2);
```

```
                        r2s=r2s+r2;
                              }
                        if(r2s<4*a) {
                        r2s=sqrt(r2s);
                        double ff= (2*a-r2s)/r2s;
                    for(di=0;di<2;di++){
                        fH[di]=fH[di]+ff*xd[di]; // change in force by using du/dx
                        f[di][J]=f[di][J]-ff*xd[di];
                              }
                        }
                        J=list[J];
                        r2s=0;
                }
               //niegbouring cells
                int jcell0=4*(i-1);
//looking for all the neighboring cells with present head of cell chosen
                for(int nabor=1;nabor<5;nabor++){
                      int jcell= map[jcell0 + nabor];
                       J = head[jcell];
                       while(J!=0){
                       for(di=0;di<2;di++){
                       xd[di]=cordH[di]-cord[di][J];
                       xd[di]=xd[di]-box*round(double(xd[di]/box));
                       r2=pow(xd[di],2);
                       r2s=r2s+r2;
                              }
                       if(r2s<4*a) {
                         r2s=sqrt(r2s);
                         double ff= (2*a-r2s)/r2s;
                         for(di=0;di<2;di++){
                         fH[di]=fH[di]+ff*xd[di];
                         f[di][J]=f[di][J]-ff*xd[di];
                              }
                        }
                              J= list[J];
                              r2s=0;
                                }
                        }
                          f[0][I]=fH[0];
                          f[1][I]=fH[1];
                          I= list[I];
                      }
              }
}
```

A.1 Simulating the variable speed Vicsek model

Note: A Constant speed model can also be simulated with the same code by putting the $\gamma = 0$

```
#include<iostream>
#include<stdlib.h>
#include<time.h>
#include<stdio.h>
#include<math.h>
#include<fstream>
using namespace std;
double** cord;
double** cordp;
double** theta;
double** f;
double** thetao;
int* mapv;
int* list;
int* head;
double* tot;
double round(double a){
return floor(a+.5);
}
const int N=2000,N1=N+1;        //N number of the particles in the system
const double a=1,rc=2,rC2=rc*rc, phi=8.61,box=sqrt(N*3.142*rc*rc/phi)..
,d=2*a,box1=box+1,vm=.1;
const int nc=floor(box1/rc),nc1=nc+1,NC=4*ceil(box/d),Nt=NC+N1,gamma=6;
const int nc2=nc*nc,nc21=nc2+1,MV=8*nc2,M=4*nc2,Nt1= N1;
int i,j,t=0,k,n=0,di,input,counti;
double xd[2],r2,r2s,dt=1, nL=0.625,thetafc,thetafs,cordH[2],fH[2];

** define various subroutine required for the main program **

int main(){
        int check;
time_t rawtime;
  struct tm * timeinfo;
         //memory allocation for different arrays
          srand((unsigned)time(0));
      ofstream krit("File Directory to write data");
      theta = (double**) malloc(2 * sizeof(double*));
          thetao = (double**) malloc(2 * sizeof(double*));
          cord = (double**) malloc(2 * sizeof(double* ));
          cordp = (double**) malloc(2 * sizeof(double* ));
          mapv= (int*) malloc(MV* sizeof(int));
          tot = (double*) malloc(N1* sizeof(double));
         head=(int*) malloc(nc21* sizeof(int));
```

```
44          f = (double**) malloc(2 * sizeof(double* ));
45          list = (int*) malloc(Nt1* sizeof(int));
46          for(i=0;i<2;i++)
47                 {
48                 theta[i] = (double*)malloc(N1 * sizeof(double));
49                 }
50                  for(i=0;i<2;i++)
51                 {
52                 thetao[i] = (double*)malloc(N1 * sizeof(double));
53                 }
54          for(i=0;i<2;i++)
55                 {
56                 f[i] = (double*)malloc(Nt1 * sizeof(double));
57                 }
58          for(i=0;i<2;i++)
59                 {
60                 cord[i] = (double*)malloc(Nt1 * sizeof(double));
61                 }
62                 mapev();        // deciding the neighbors
63                  for(i=1;i<N1;i++){
64                  double f=3.142*2*((rand()))/(double)RAND_MAX;
65                  theta[0][i]=cos(f); theta[1][i]=sin(f);
66                  thetao[0][i]=theta[0][i]; thetao[1][i]=theta[1][i];
67           }
68      input=2000;
69           time (&rawtime);
70   timeinfo = localtime (&rawtime);
71   cout<<asctime(timeinfo);
72 for(i=1;i<N1;i++) tot[i]=1;
73   int Sq= sqrt(N)+1;
74   for(i=0;i<Sq;i++){
75   for(j=0;j<Sq;j++){
76      if(n>N1) break;
77       cord[0][n]=(i+.3)*(box-rc)/(Sq);
78       cord[1][n]=(j+.3)*(box-rc)/(Sq);
79       cord[0][n]-=box*round(cord[0][n]/(box));
80       cord[1][n]-=box*round(cord[1][n]/(box));
81       n++;
82                          }
83                 }
84           t=0;
85         while(t<input){
86      link();
87    for(i=1;i<N1;i++){
88            cord[0][i] += vm*(theta[0][i])*tot[i]*dt;
89            cord[0][i] -=box*round(cord[0][i]/(box));
90            cord[1][i] += vm*(theta[1][i])*tot[i]*dt;
91            cord[1][i] -=box*round(cord[1][i]/(box));
```

```
92          }
93     average();
94          t++;
95           }
96    time (&rawtime);
97    timeinfo = localtime (&rawtime);
98  cout<<asctime(timeinfo);
99  }
```

A.2 Simulating the Active driven disks

Constant speed

```
1  ** Define all the variables required globally **
2  ** Define subroutines required for the main program **
3   \\subroutine for calculating the propulsion force
4  void totforce(void){
5          for(j=1;j<N1;j++){
6            f[0][j]+=Vp*cos(theta[j]);
7            f[1][j]+=Vp*sin(theta[j]);
8      \\Vp is the constant propulsion velocity
9                  }
10 }
11 int main(double){
12 ** define all memory allocations similar to variable velocity vicsek model code **
13   mape();
14          inputn=2000000;
15       while( t < inputn)
16          {
17                  link();
18          force();                        //potential generated force
19                  totforce();                     // propulsion generated force
20                  for(i=1;i<N1;i++){
21                  for(di=0;di<2;di++){            // di is looping over the dimension
22                   cord[di][i]=cord[di][i]+f[di][i]*dt;
23                   cord[di][i]-=box*round(double(cord[di][i]/box));
24                  }
25 }
26
27          for(i=1;i<N1;i++){
28            W=gasdev();
29            // gasdev is function creating Gaussian distribution
30             theta[i]+=1.414*W*Noi*dt;       // Noi is the noise strength
31                  }
32                  t++;
33          }
34           time (&rawtime);
```

```
          timeinfo = localtime (&rawtime);
          cout<<asctime(timeinfo);
}
```

Variable speed:

```
** Define all the variables required globally **
** Define subroutines required for the main program **
// In the average subroutine in the last lines add a random number generated
// by gasdev into propulsion direction using the formulas of cos(a+b), sin(a+b)
**~subroutine average
 double f = nL*gasdev();
          double temp;
          temp = theta[0][I]*cos(f)-sin(f)*theta[1][I];
          theta[1][I] = theta[1][I]*cos(f)+sin(f)*theta[0][I];
          theta[0][I]=temp;
**

int main(){
** define all memory allocations similar to variable..
 velocity vicsek model code **
** Initialize the system homogeneously or in square lattice mentioned
  variable velocity vicsek model code **
        mape();   //required four neighbors for the force calculation
        mapev();  //required 8 neighbors for the aligning interactions

for(i=1;i<N1;i++) tot[i]=1;
          cin>>input;
           time (&rawtime);
  timeinfo = localtime (&rawtime);
  cout<<asctime(timeinfo);
          t=0;
        while(t<input){
     link();
          force();
          average();
   for(i=1;i<N1;i++){
            cord[0][i] += vm*(theta[0][i])*tot[i]*dt+f[0][i]*dt;
            cord[0][i] -=box*round(cord[0][i]/(box));
            cord[1][i] += vm*(theta[1][i])*tot[i]*dt+f[1][i]*dt;
            cord[1][i] -=box*round(cord[1][i]/(box));
        }
          time (&rawtime);
  timeinfo = localtime (&rawtime);
cout<<asctime(timeinfo);
}
```

Bibliography

[1] Tams Vicsek, Anna Zafeiris, Physics Reports 517 (2012) 71140

[2] Robert Zwanzig, *Nonequilibrium Statistical Mechanics*, Oxford university Press, (2001).

[3] P. M. Chaikin, T. C. Lubensky *Principles of Condensed Matter Physics* , Cambridge University Press ,(1995)

[4] Daan Frenkel, Berend Smit, *Understanding Molecular Simulation, From Algorithms to Applications*, Academic Press, (1996).

[5] Narayan, V., S. Ramaswamy, and N. Menon, 2007, Science 317,105.

[6] Kemkemer, R., D. Kling, D. Kaufmann, and H. Gruler, 2000, Eur.Phys. J. E 1, 215.

[7] Frank Schweitzer *Brownian Agents and Active Particles*,Springer Series in Synergetics,(2003).

[8] Surrey, T., F. J. Nedelec, S. Leibler, and E. Karsenti, 2001, Science 292, 1167.

[9] Vicsek, T., A. Czirok, E. Ben-Jacob, I. Cohen, and O. Shochet,1995, Phys. Rev. Lett. 75, 1226.

[10] J. Toner, Y. Tu and S. Ramaswamy, Ann. Phys. 318 (2005) 170.

[11] M. C. Marchetti, J. F. Joanny,S. Ramaswamy, T. B. Liverpool, J. Prost, Madan Rao, R. Aditi Simha,Reviews of Modern Physics,85, 3, 2013.

[12] Shradha Mishra, Kolbjrn Tunstrm, Iain D. Couzin, and Cristin Huepe,. Phys. Rev. E 86, 011901, 2012

[13] Katz, Y. et al. PNAS, 108 46, 18720-18725.

[14] Fily Y1, Marchetti MC.Phys Rev Lett. 2012 23,1082,2012

[15] Gabriel S. Redner, Michael F. Hagan, and Aparna Baskaran,PRL 110, 055701 (2013)

[16] B. M. Mognetti,A. Saric, S. Angioletti-Uberti, A. Cacciuto, C. Valeriani, and D. Frenkel,PRL 111, 245702 (2013)

[17] Saul Teukolsky, Brian P. Flannery, William T. Vetterling, and William H. Press *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 2007