Homework 1

Simulation of complex systems FFR120

CHALMERS
UNIVERSITY OF TECHNOLOGY

Nicole Adamah

November 24, 2022

Contents

1 Exercise 8.4

Vicsek model at low noise and low density.
In this exercise, take L=100,N=100,v=1,t = 1,and = 0.01.Implement your code and set the number of iterations to S = 104 steps. Compare your results with figure 8.3.
a. Generate an initial particle configuration and save it in order to be able to start from exactly the same configuration for different flocking radii Rf . Plot the initial configuration.

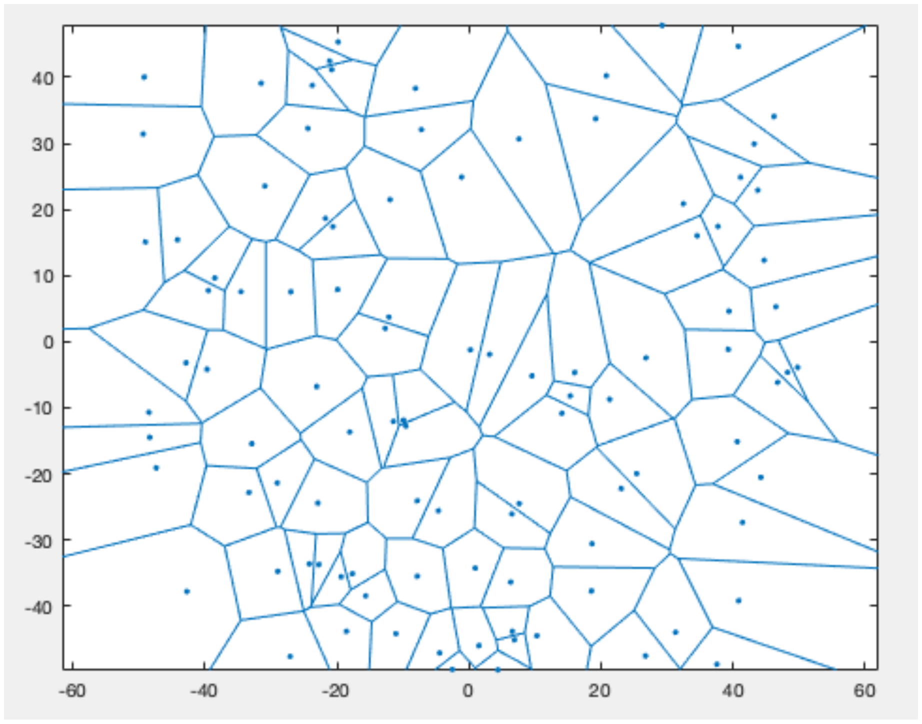

Figure 1: *Initial Voronoi plot*

b. Perform the simulation for Rf = 1. Calculate and plot the global alignment and clustering coefficients as a function of the time step.

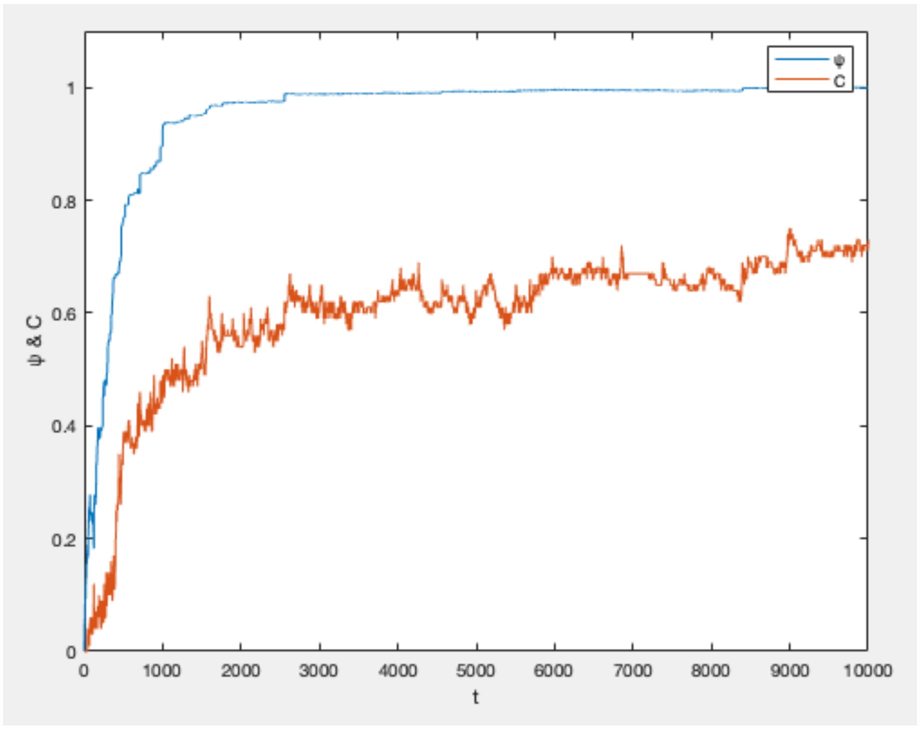

Figure 2: *Plot of the global alignment and clustering coefficients as a function of the time step, with Rf = 1*

c. Plot the configurations after 10, 100, 500, and 1000 iterations, as well as the final configuration.

-The amount of clusters increases with increased iterations. This can also be seen in figure 2 where A and C converges towards 1 which indicates high orientational order respectively high level of clustering.Argun et al., 2021.

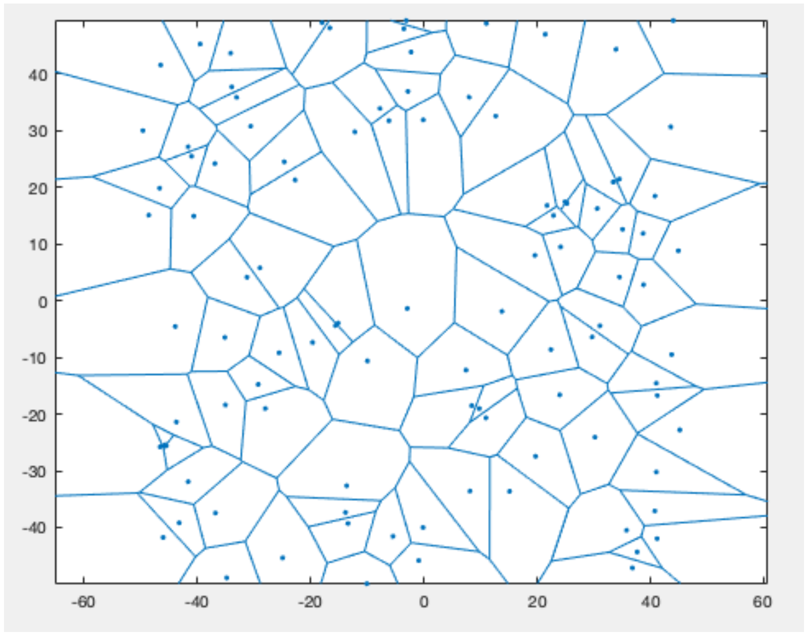

Figure 3: *Plot of configuration after 10 iterations*

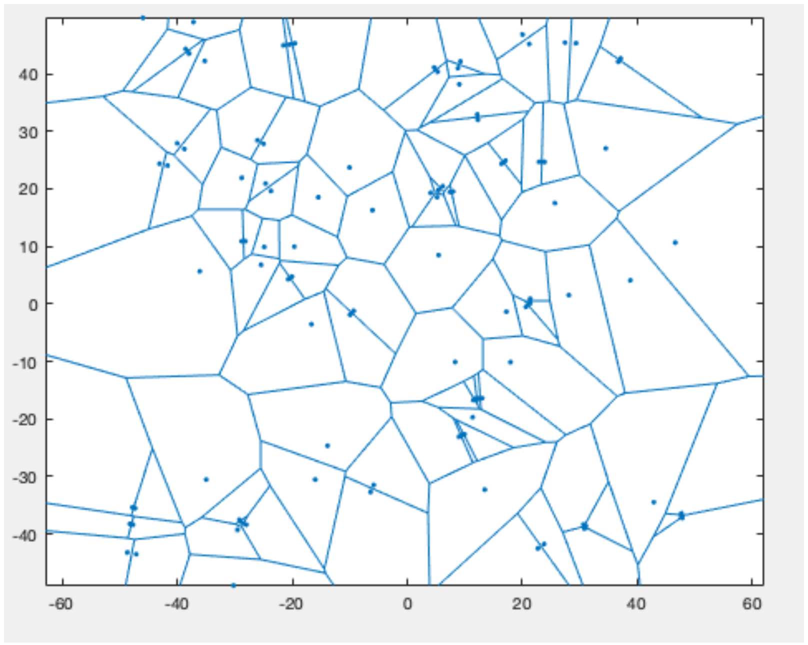

Figure 4: *Plot of configuration after 100 iterations*

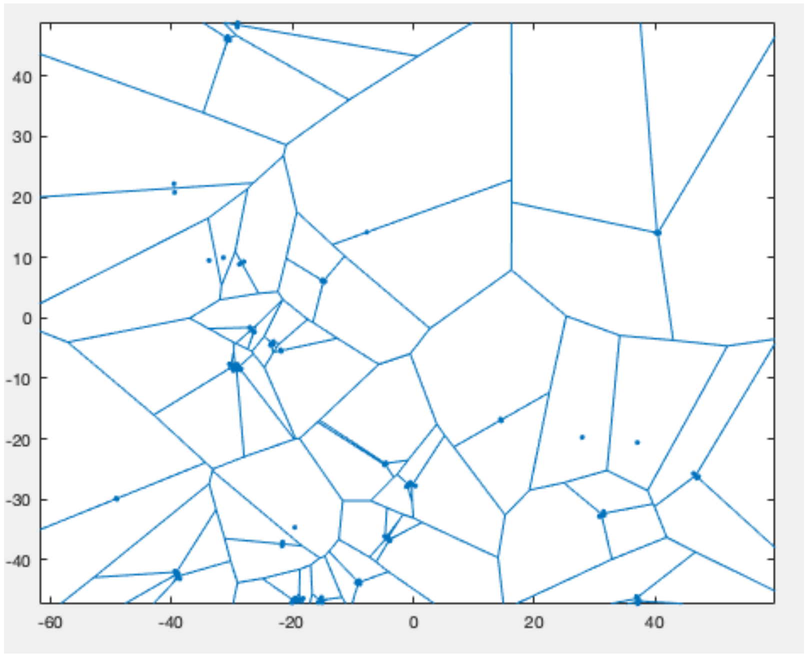

Figure 5: *Plot of configuration after 500 iterations*

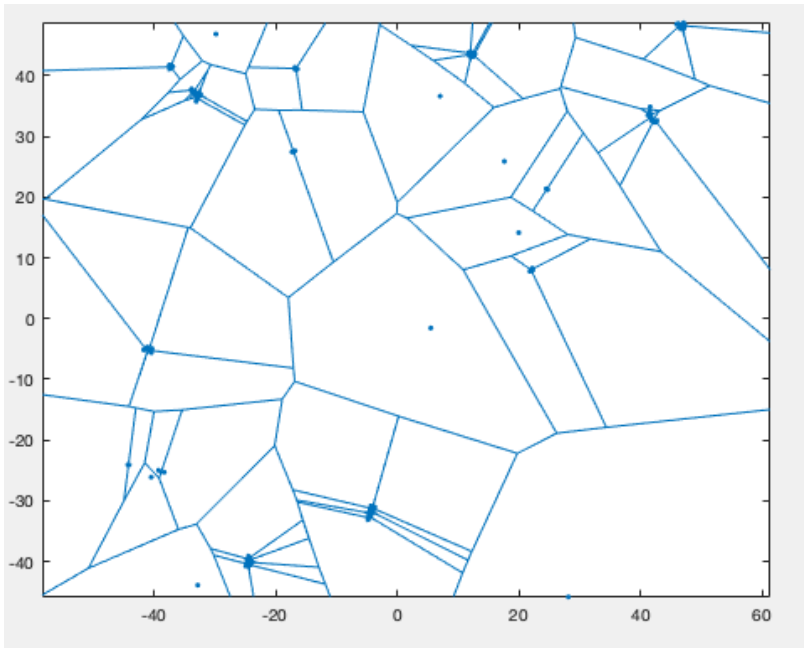

Figure 6: *Plot of configuration after 1000 iterations*

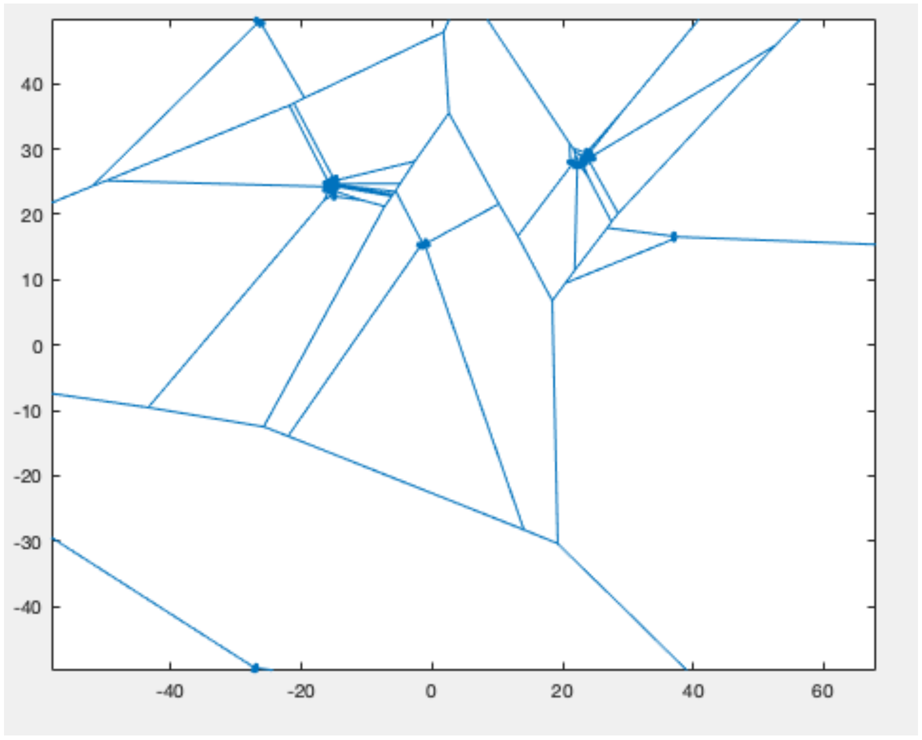

Figure 7: *Voronoi plot after 10 000 iterations with Rf = 1*

d. Repeat the simulation starting from the same initial configuration, but taking a larger detection radius (e.g., Rf = 2, 5, 10, ...).
-When increasing the value on Rf the particles, when clustering, tend to remain farther apart from each other. This is due to the alignment mechanism: if Rf is larger, particles start to align with particles further away.Argun et al., 2021.

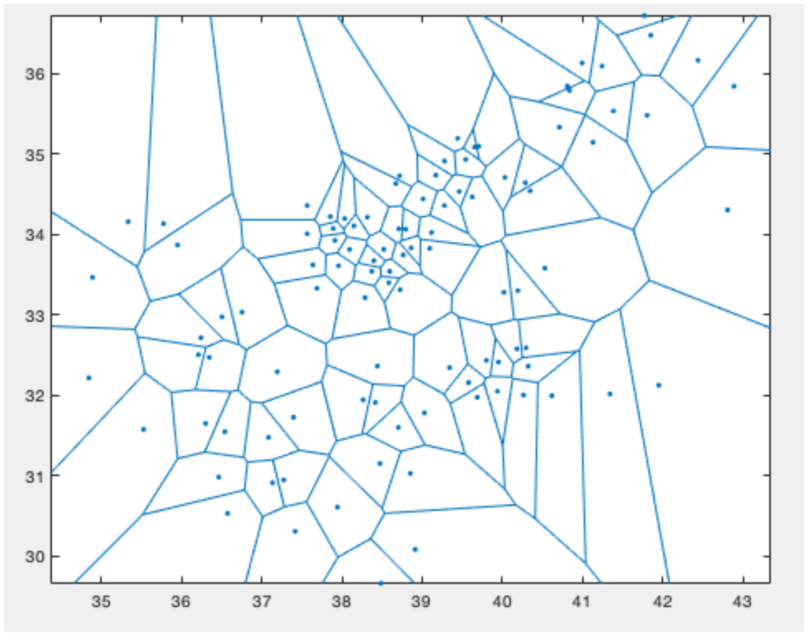

Figure 8: *Plot of configuration after 10000 iterations with Rf=2*

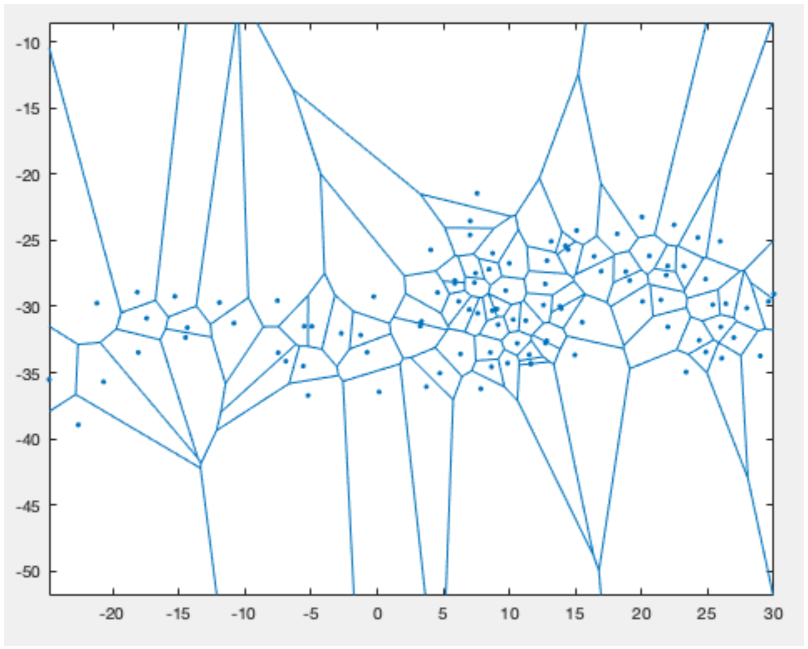

Figure 9: *Plot of configuration after 10000 iterations with Rf=5*

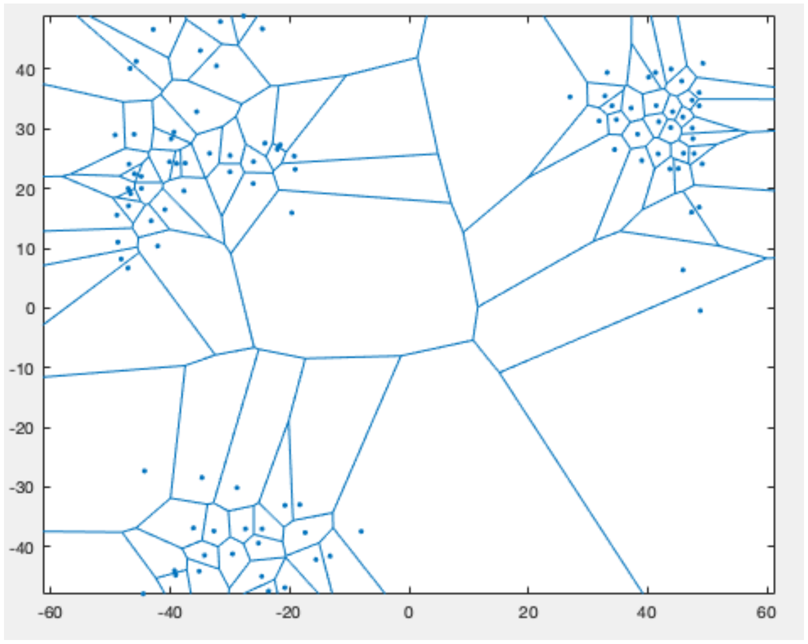

Figure 10: *Plot of configuration after 10000 iterations with Rf=10*

2 Exercise 8.5

Increased noise. *Repeat exercise 8.4 using a increased noise level, for example, = 0.1. For ease of comparison, keep the other parameters the same. Run your simulation for S = 104 steps. Compare your results with figure 8.4.*

- With increased noise, consequently the particles are slightly less aligned and slightly worse organized in clustersArgun et al., 2021.The noise makes the clusters smaller and less stable which can be seen in the clustering coefficient in figure 12.

a. Generate an initial particle configuration and save it in order to be able to start from exactly the same configuration for different flocking radii Rf . Plot the initial configuration.

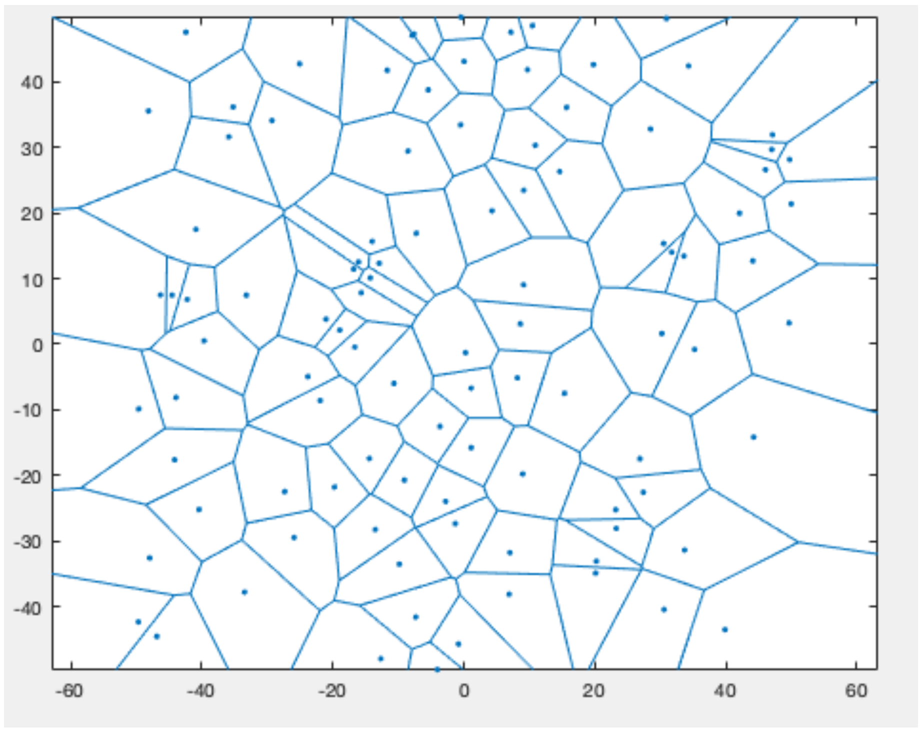

Figure 11: *Initial Voronoi plot with eta = 0.1*

b. Perform the simulation for Rf = 1. Calculate and plot the global alignment and clustering coefficients as a function of the time step.

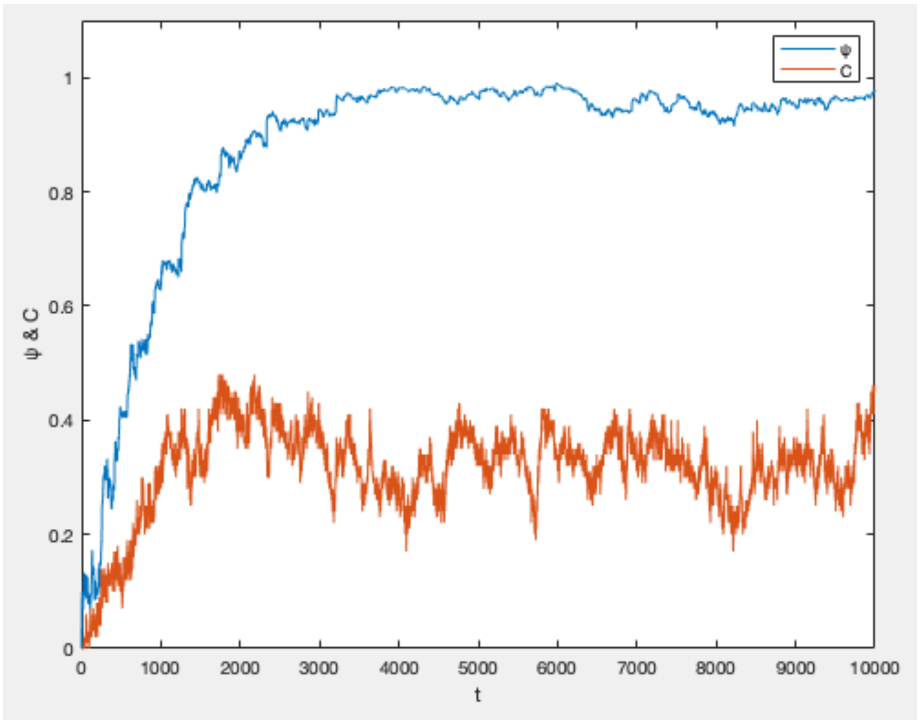

Figure 12: *Plot of the global alignment and clustering coefficients as a function of the time step, with Rf = 1 and eta = 0.1*

c. Plot the configurations after 10, 100, 500, and 1000 iterations, as well as the final configuration.

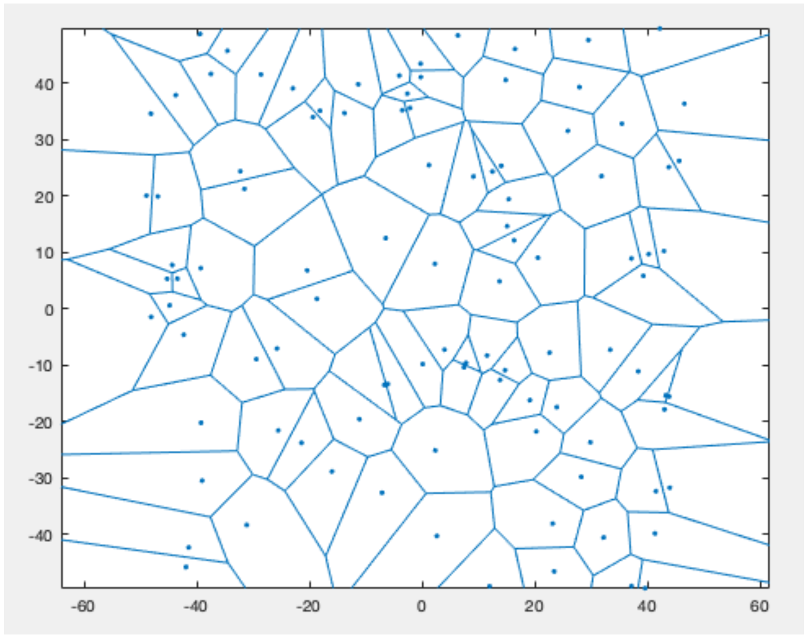

Figure 13: *Plot of configuration after 10 iterations with eta = 0.1*

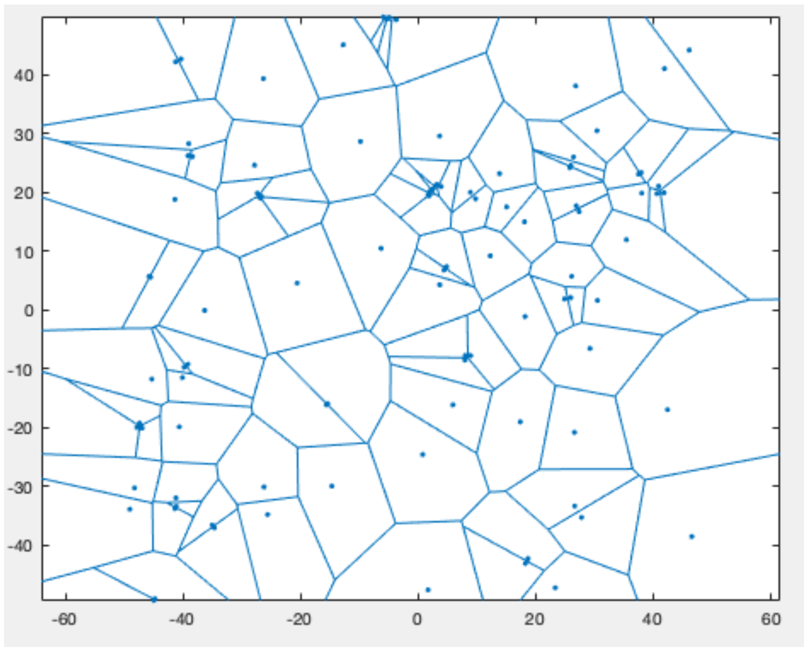

Figure 14: *Plot of configuration after 100 iterations with eta = 0.1*

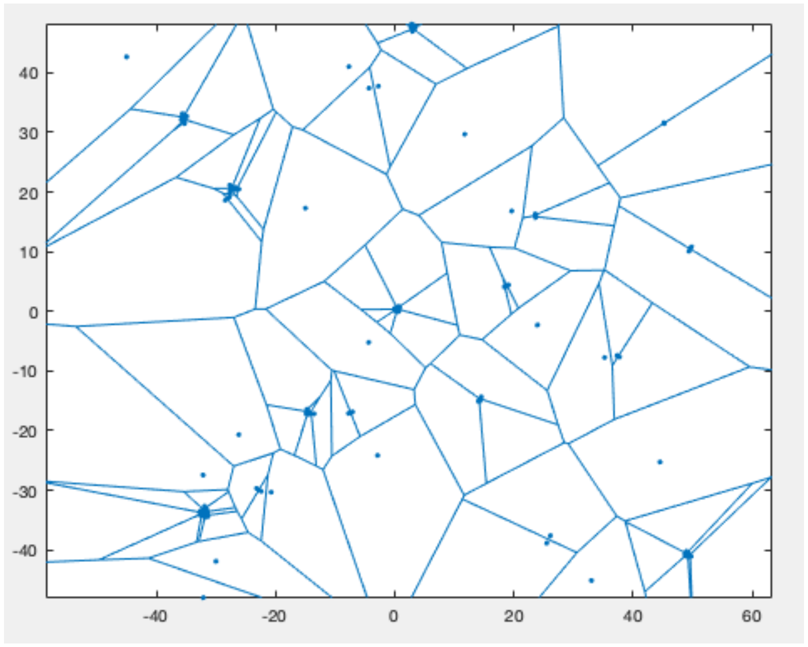

Figure 15: *Plot of configuration after 500 iterations with eta = 0.1*

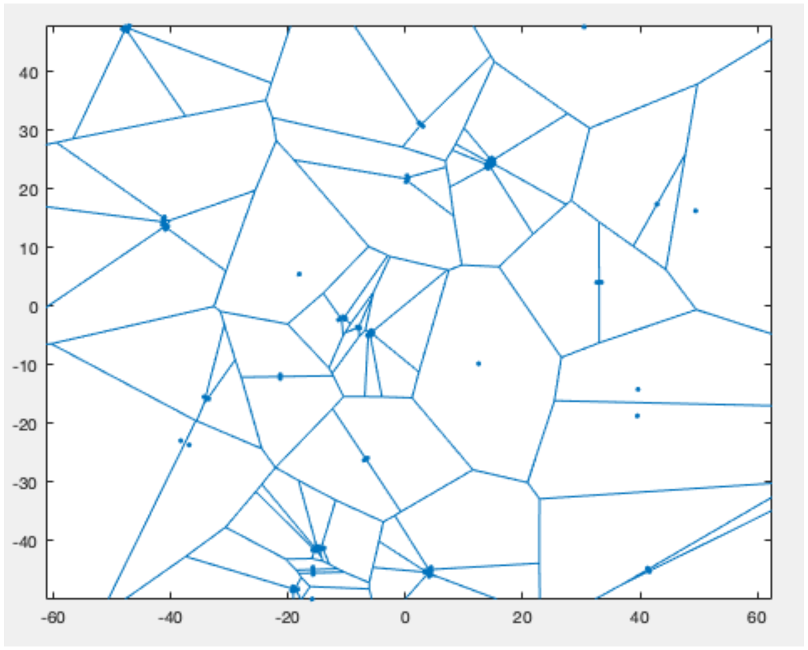

Figure 16: *Plot of configuration after 1000 iterations with eta = 0.1*

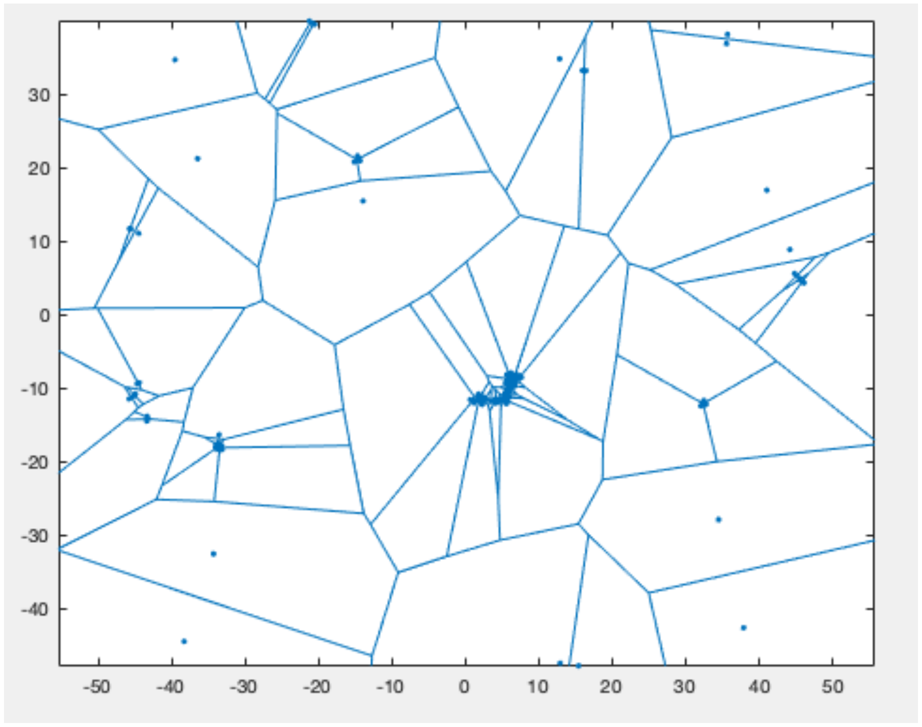

Figure 17: *Voronoi plot after 10000 iterations with Rf = 1 and eta = 0.1*

d. Repeat the simulation starting from the same initial configuration, but taking a larger detection radius (e.g., Rf = 2, 5, 10, ...).

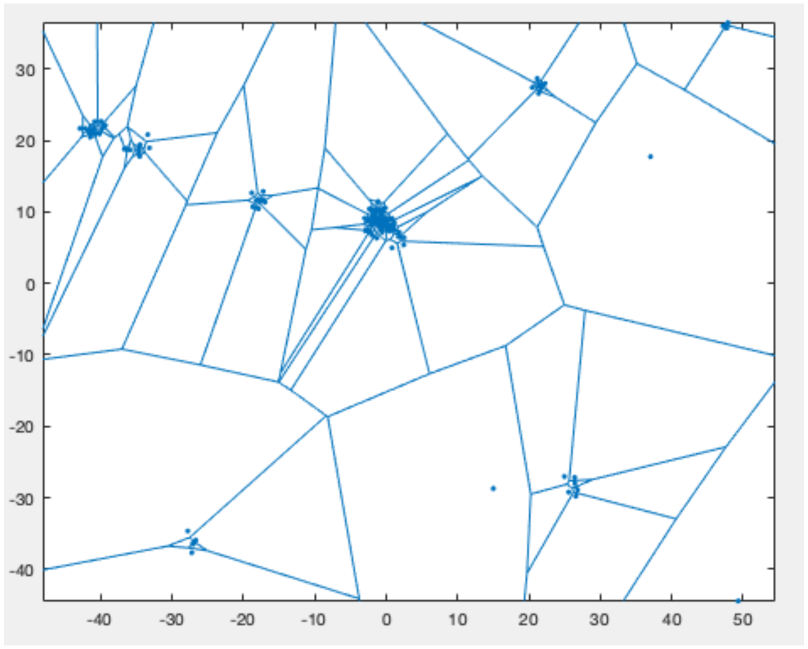

Figure 18: *Plot of configuration after 10000 iterations with Rf=2*

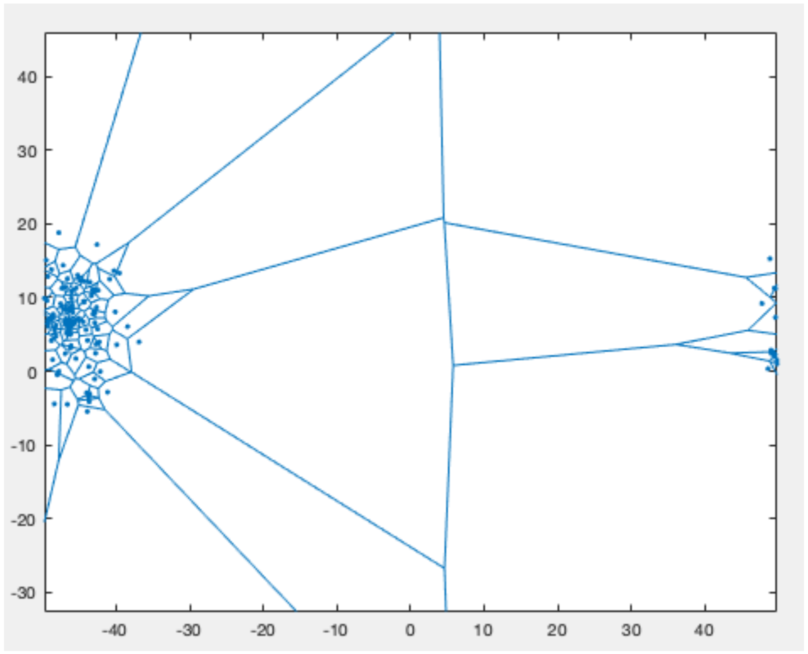

Figure 19: *Plot of configuration after 10000 iterations with Rf=5*

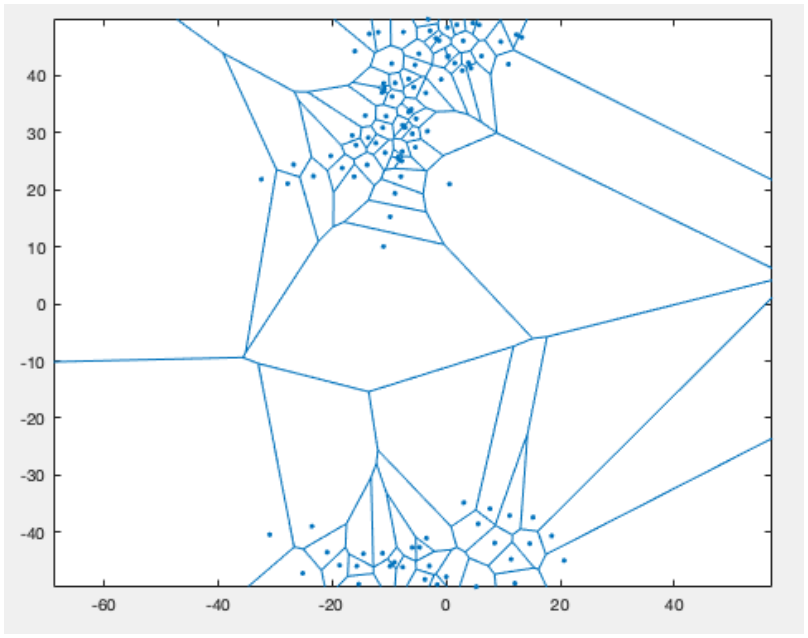

Figure 20: *Plot of configuration after 10000 iterations with Rf=10*

3 Exercise 8.6

Higher density. *Repeat exercise 8.4 using a higher density of particles, for example, N = 1000 (keep the other parameters the same). Run your simulation for S = 10 000 steps. Compare your results with figure*

-With higher density the particles rapidly show a phase with a high orientational order, see figure 22 where the alignment coefficient converges to 1 fast. For Rf = 1 the particles organize into several stable clusters compared to when Rf = 5 which results in a single giant cluster including all the particles.

a. Generate an initial particle configuration and save it in order to be able to start from exactly the same configuration for different flocking radii Rf . Plot the initial configuration.

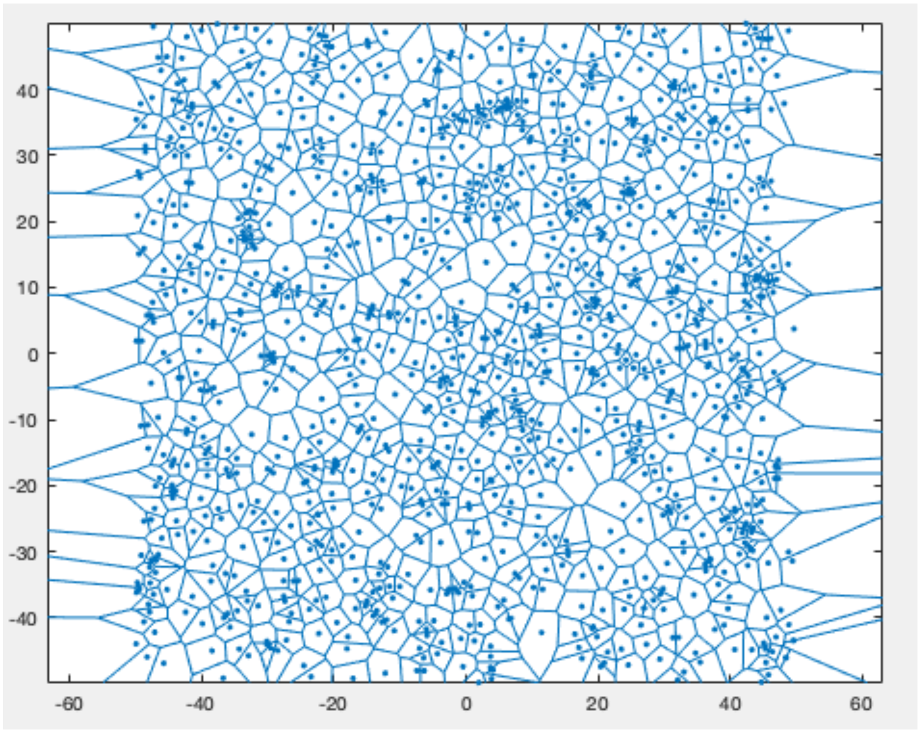

Figure 21: *Initial Voronoi plot with N = 1000*

b. Perform the simulation for Rf = 1. Calculate and plot the global alignment and clustering coefficients as a function of the time step.

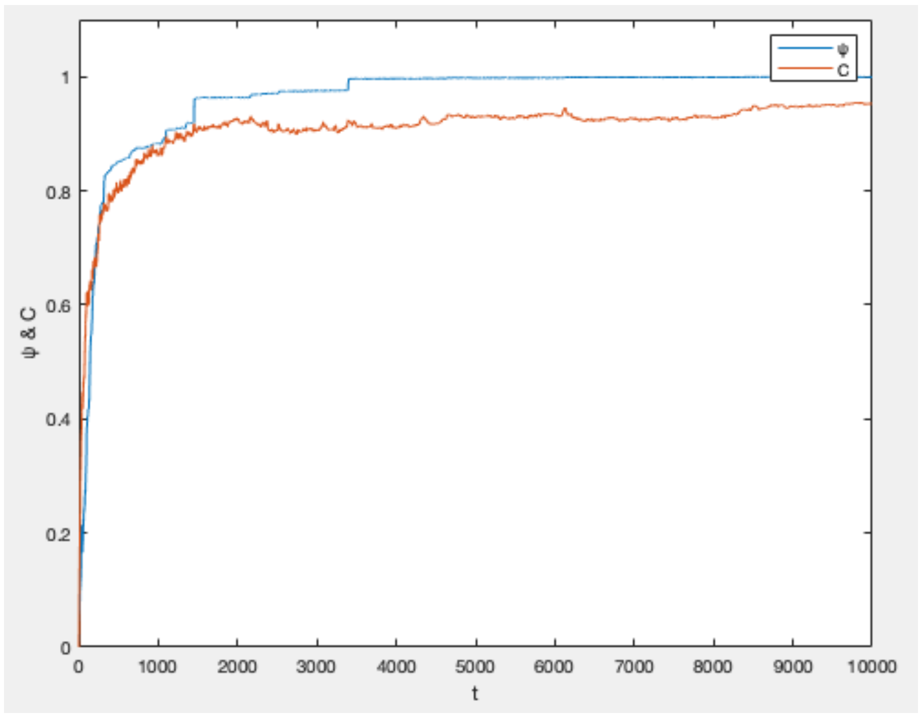

Figure 22: *Plot of the global alignment and clustering coefficients as a function of the time step, with Rf = 1 and N = 1000*

c. Plot the configurations after 10, 100, 500, and 1000 iterations, as well as the final configuration.

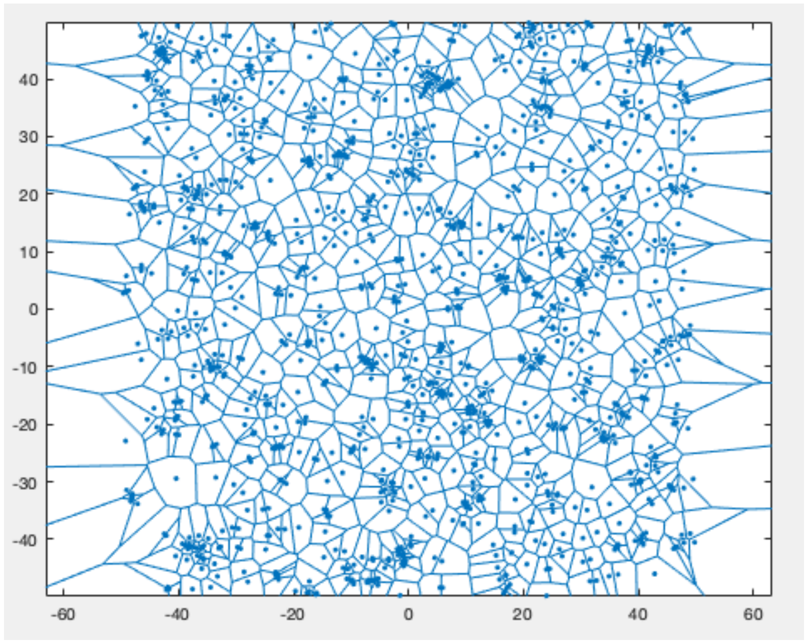

Figure 23: *Plot of configuration after 10 iterations with N=1000*

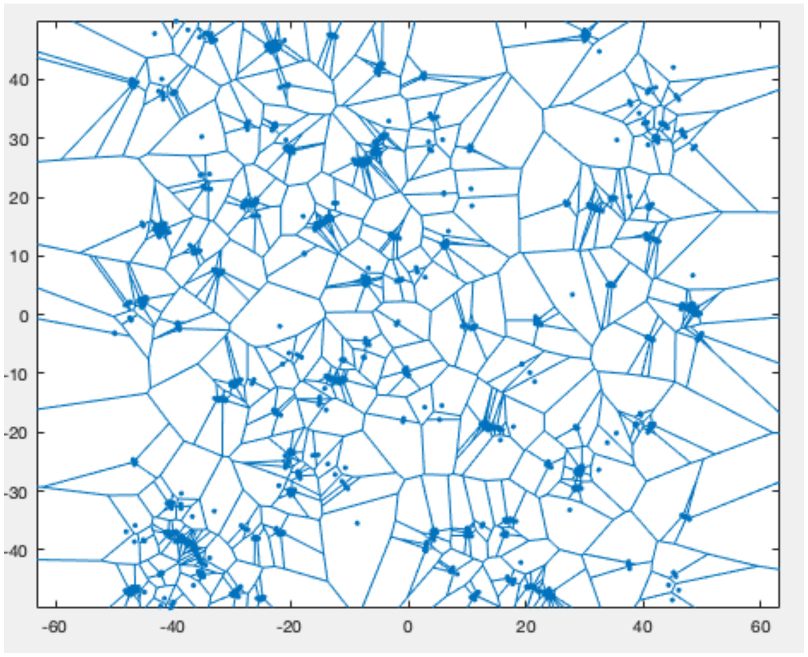

Figure 24: *Plot of configuration after 100 iterations with N = 1000*

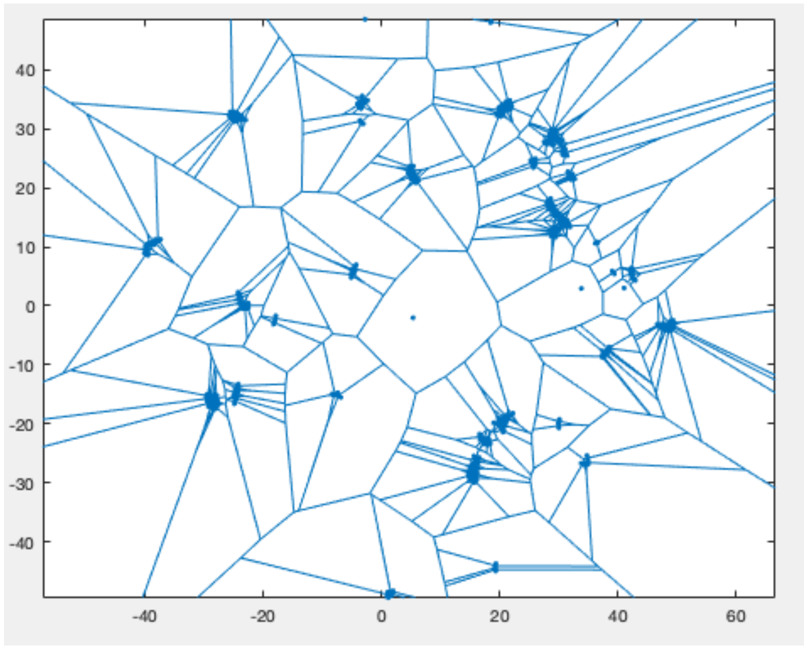

Figure 25: *Plot of configuration after 500 iterations with N = 1000*

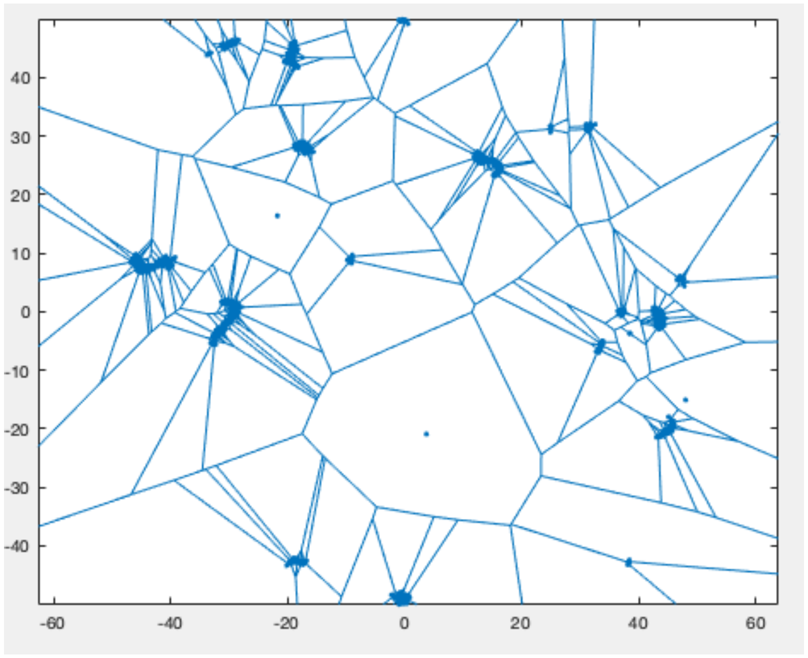

Figure 26: *Plot of configuration after 1000 iterations with N = 1000*

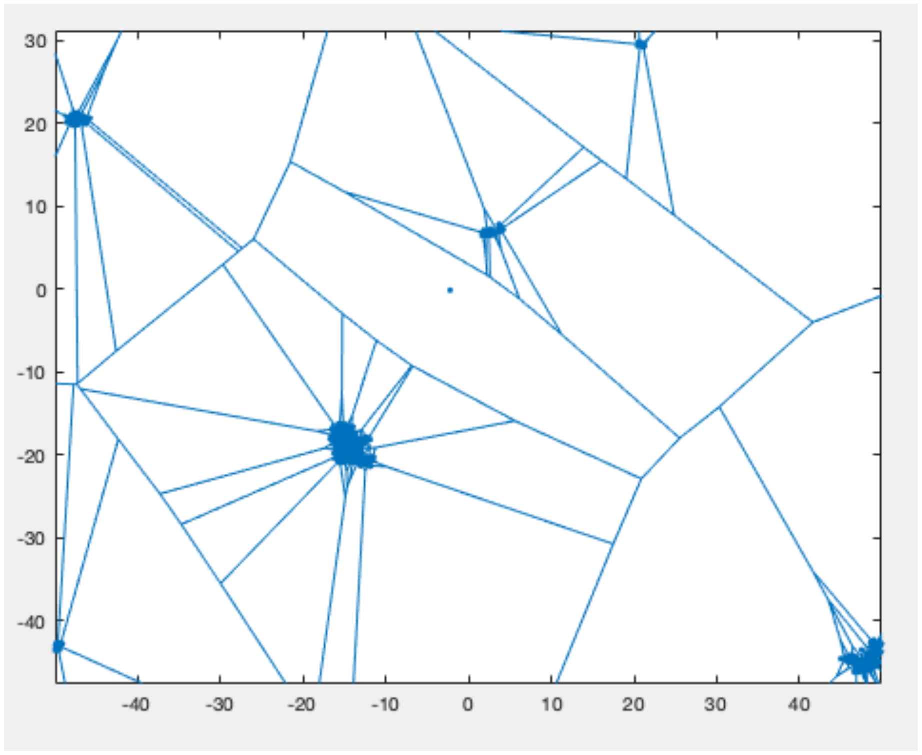

Figure 27: *Voronoi plot after 10000 iterations with Rf = 1 and N = 1000*

d. Repeat the simulation starting from the same initial configuration, but taking a larger detection radius (e.g., Rf = 2, 5, 10, ...).

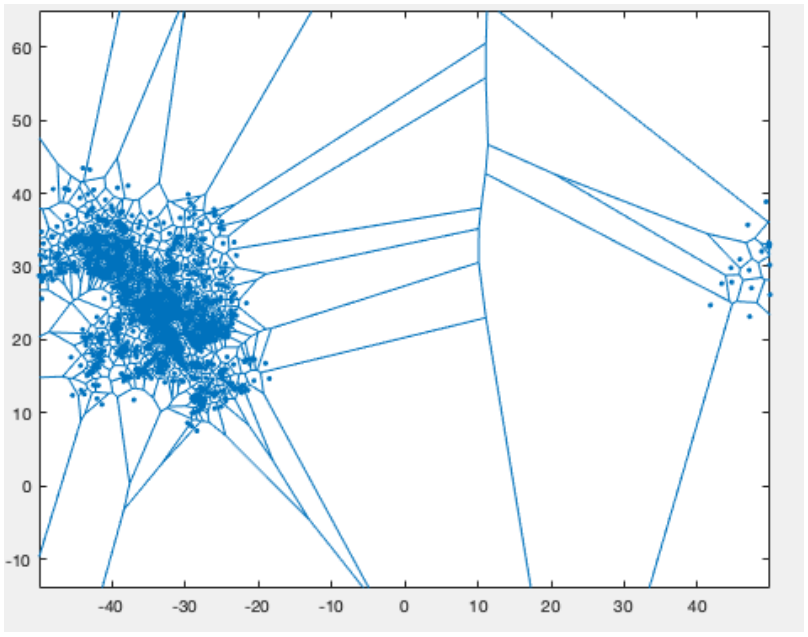

Figure 28: *Plot of configuration after 10000 iterations with Rf=5*

4 Exercise 8.7

Higher density and increased noise.
Now repeat exercise 8.4, increasing both the density (N = 1000) and the noise level (= 0.1). Run your simulation and compare your results with figure. **a. Generate an initial particle configuration and save it in order to be able to start from exactly the same configuration for different flocking radii Rf . Plot the initial configuration.**

-We can see that with high density and high noise the performance decreases when compared to simulation with only high density.

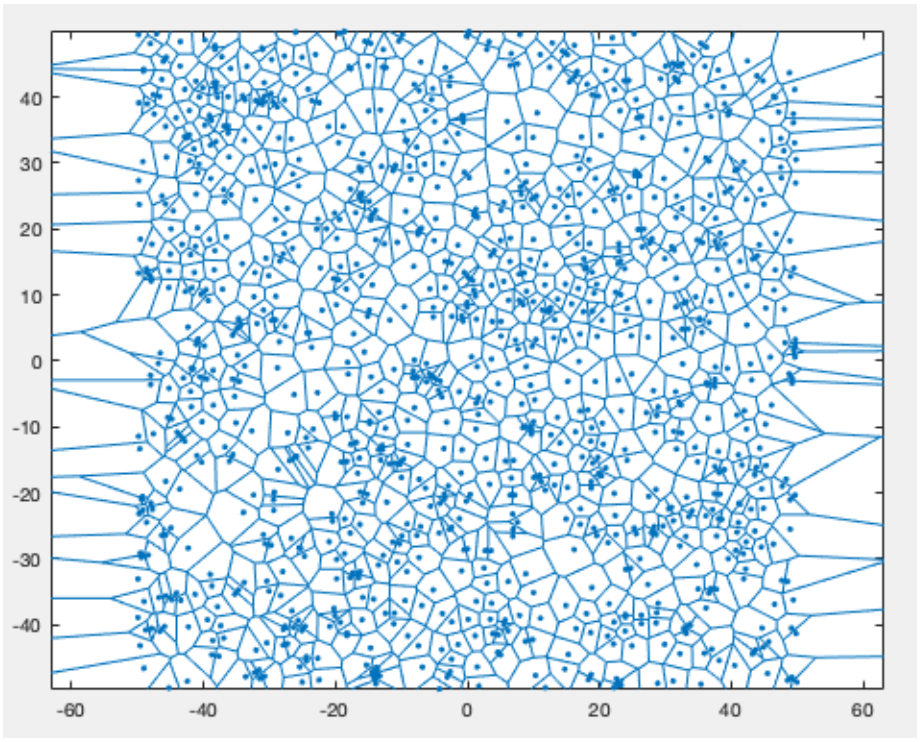

Figure 29: *Initial Voronoi plot with N = 1000 with eta = 0.1*

b. Perform the simulation for Rf = 1. Calculate and plot the global alignment and clustering coefficients as a function of the time step.

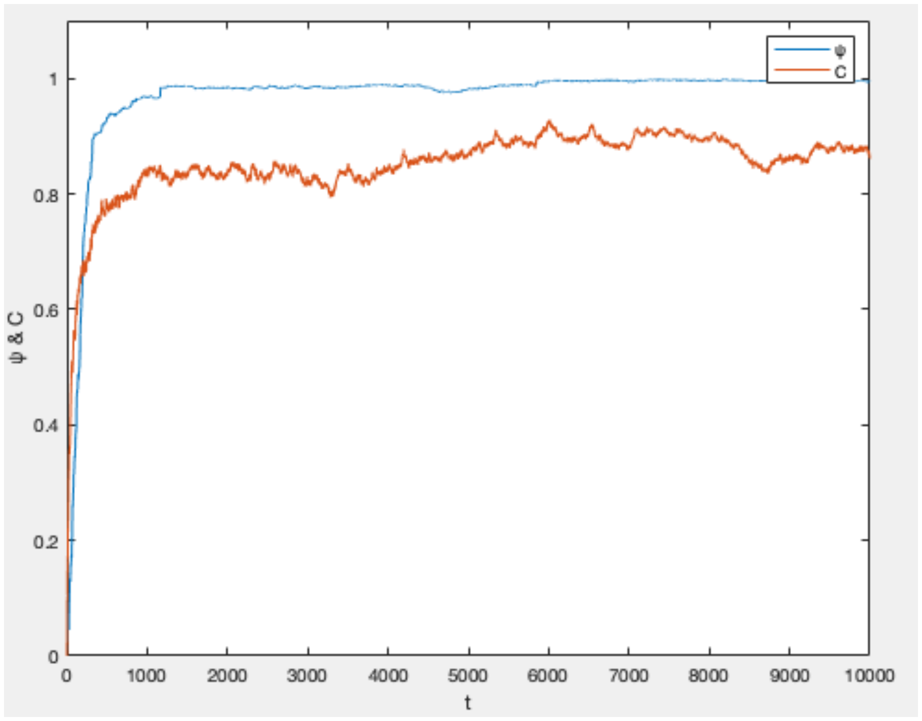

Figure 30: *Plot of the global alignment and clustering coefficients as a function of the time step, with eta = 0.1 and N = 1000*

c. Plot the configurations after 10, 100, 500, and 1000 iterations, as well as the final configuration.

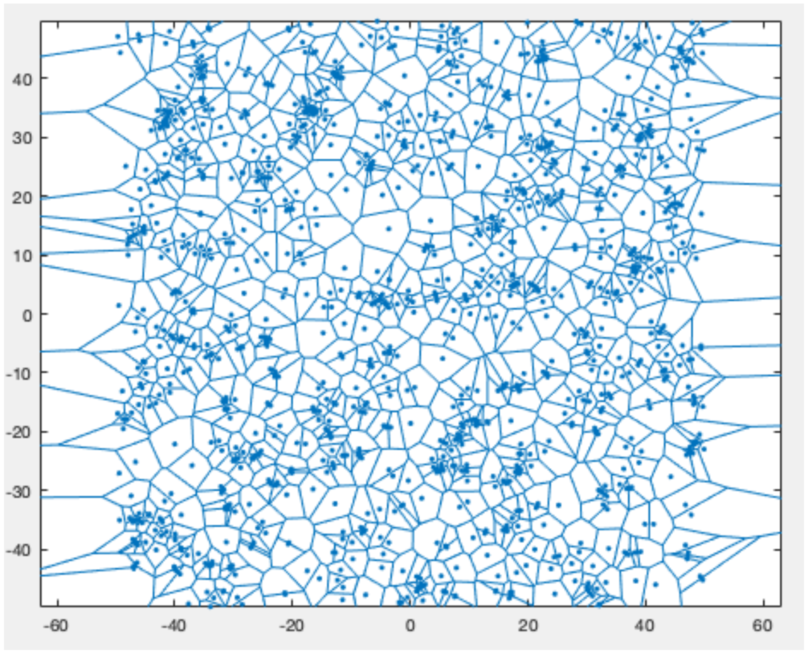

Figure 31: *Plot of configuration after 10 iterations with eta = 0.1 and N=1000*

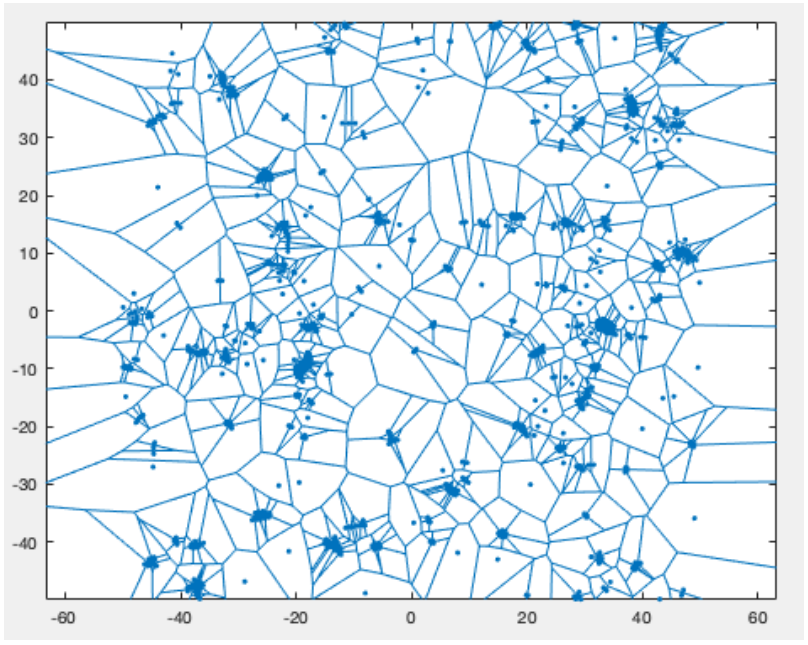

Figure 32: *Plot of configuration after 100 iterations with eta = 0.1 and N = 1000*

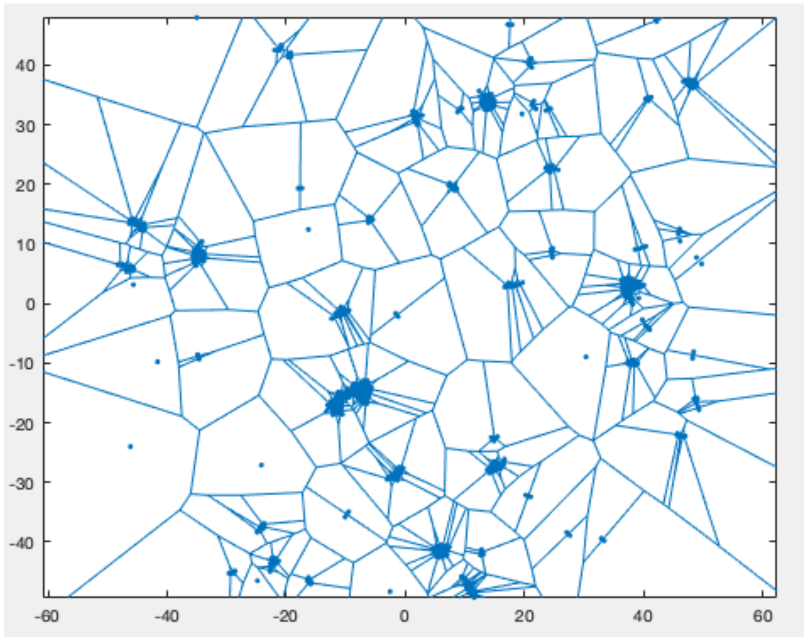

Figure 33: *Plot of configuration after 500 iterations with eta = 0.1 and N = 1000*

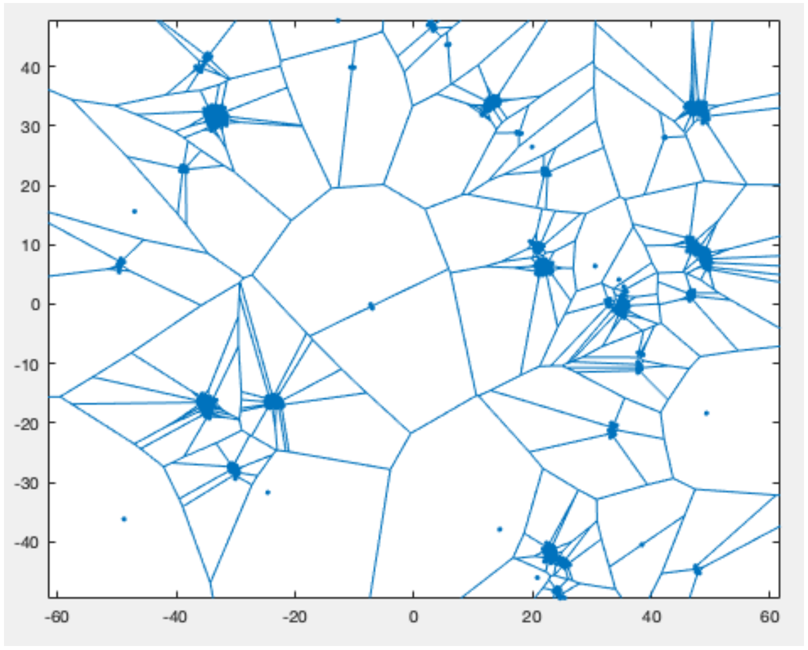

Figure 34: *Plot of configuration after 1000 iterations with eta = 0.1 and N = 1000*

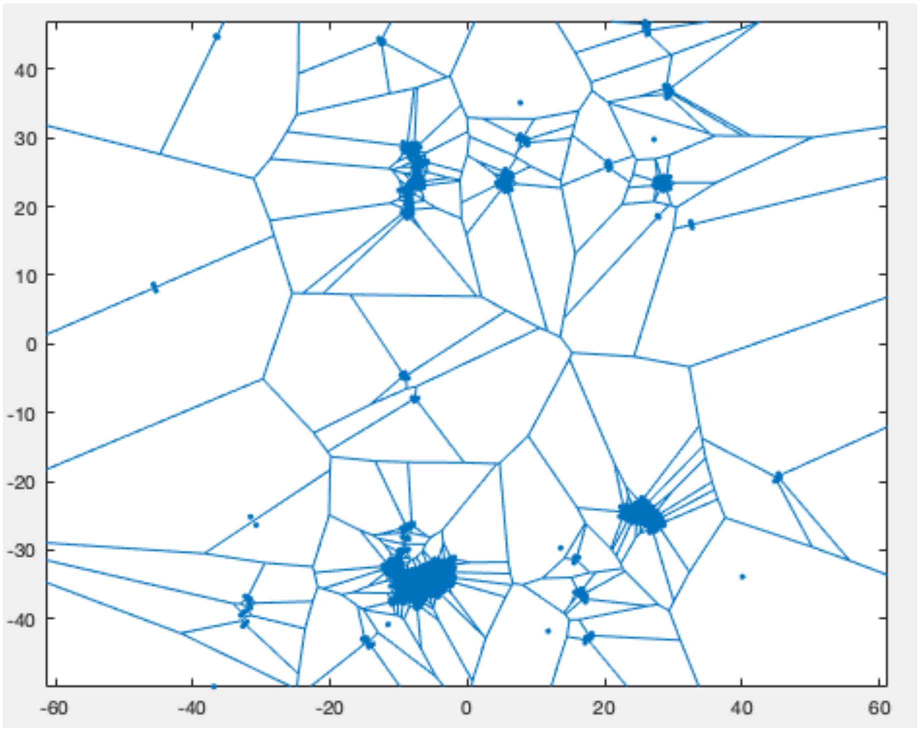

Figure 35: *Voronoi plot after 10000 iterations with eta = 0.1 and N = 1000*

d. Repeat the simulation starting from the same initial configuration, but taking a larger detection radius (e.g., Rf = 2, 5, 10, ...).

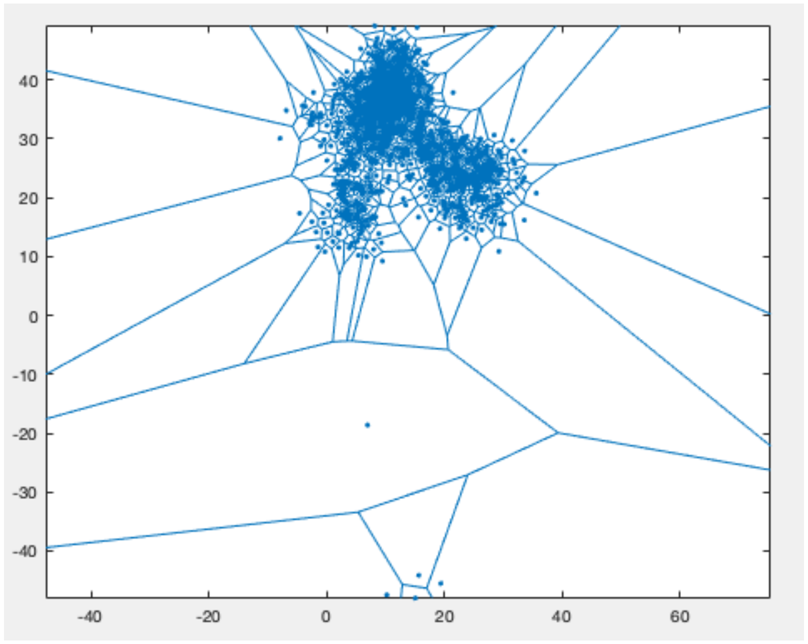

Figure 36: *Plot of configuration after 10000 iterations with Rf=5*

5 Exercise 8.10

Interaction with k nearest neighbors.
Repeat exercise 8.4 by changing the interaction rule. In contrast to the standard Vicsek model, calculate the orientation by considering only the k nearest neighbors of a particle and the particle itself. Try different values of k (e.g., k = 1, 4, 8). Compare your results with figure 8.10. Verify that this interaction might be non-metric (i.e., the interaction does not only depend on the metric distance between particles) and non-reciprocal (i.e., if particle j belongs to the k nearest neighbors of particle i, particle i is not necessarily included in the k nearest neighbors of particle j).
-The clusters increases with k.

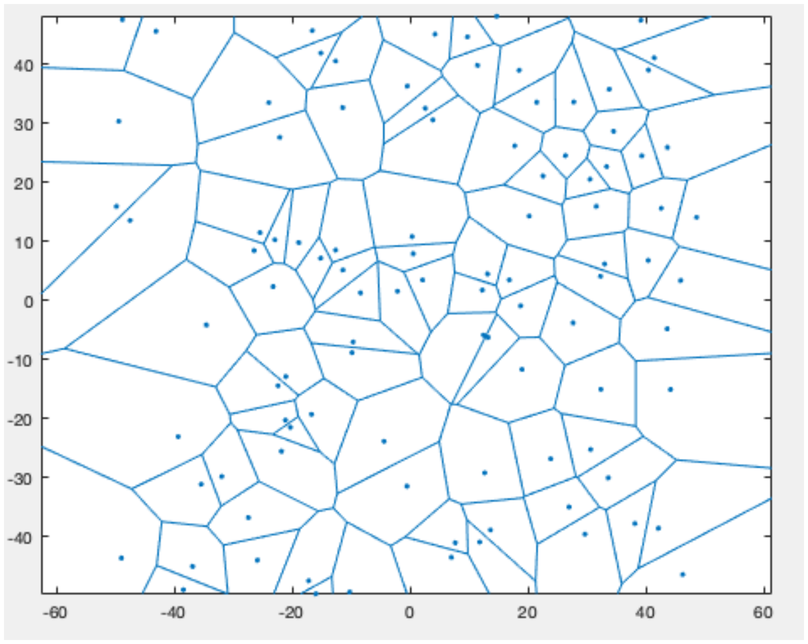

Figure 37: *Plot of configuration after 10000 iterations with KNN for k=1*

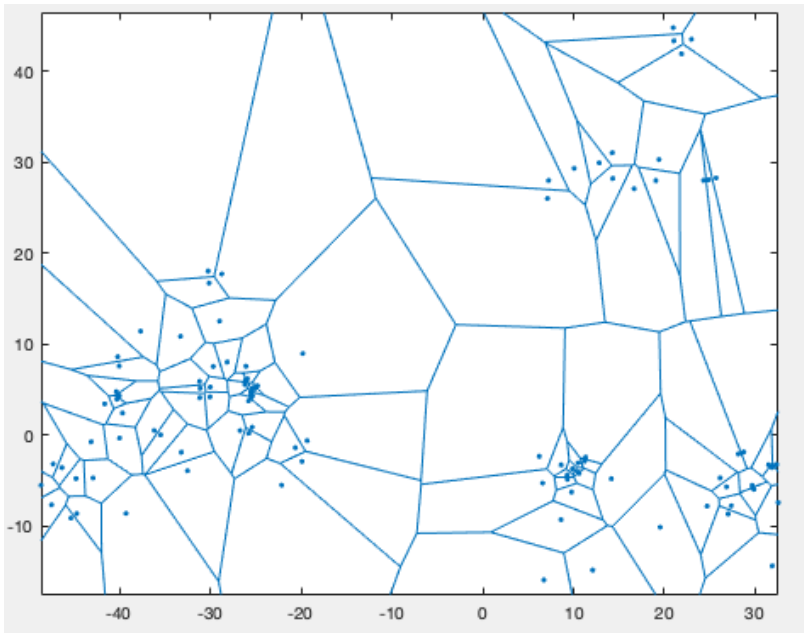

Figure 38: *Plot of configuration after 10000 iterations with KNN for k=4*

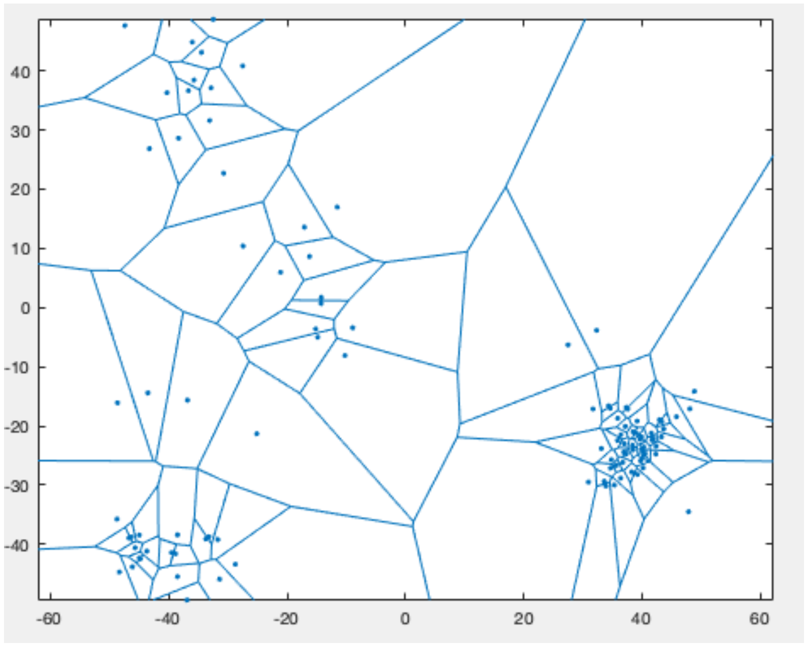

Figure 39: *Plot of configuration after 10000 iterations with KNN for k=8*

6 Exercise 8.11

Vision cones.
Restrict the field of view of the particles in exercise 8.10 so that they can only see particles within a certain vision cone (e.g., only particles in front of them). Determine how this changes the overall behavior of the system.
-The clusters decreases when implementing an angle limit.

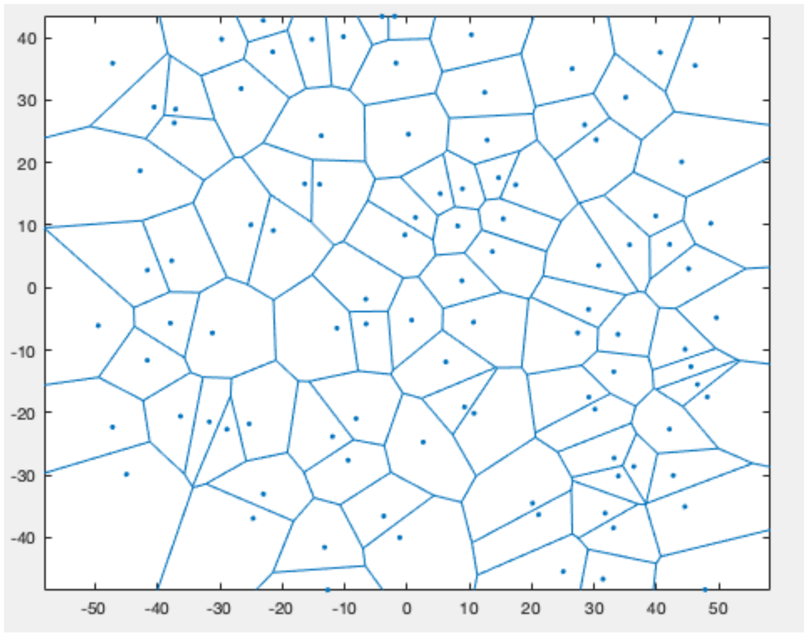

Figure 40: *Plot of configuration after 10000 iterations with KNN for k=1*

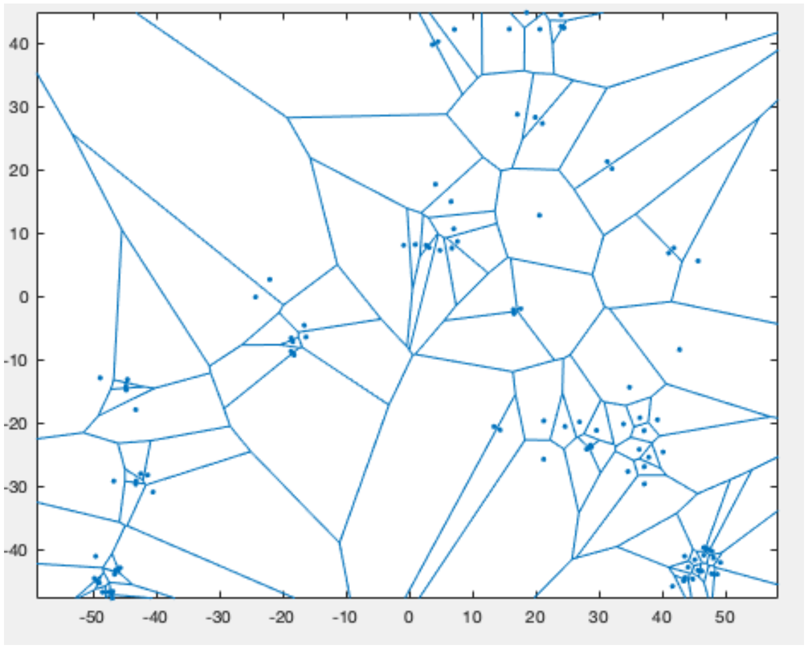

Figure 41: *Plot of configuration after 10000 iterations with KNN for k=4*

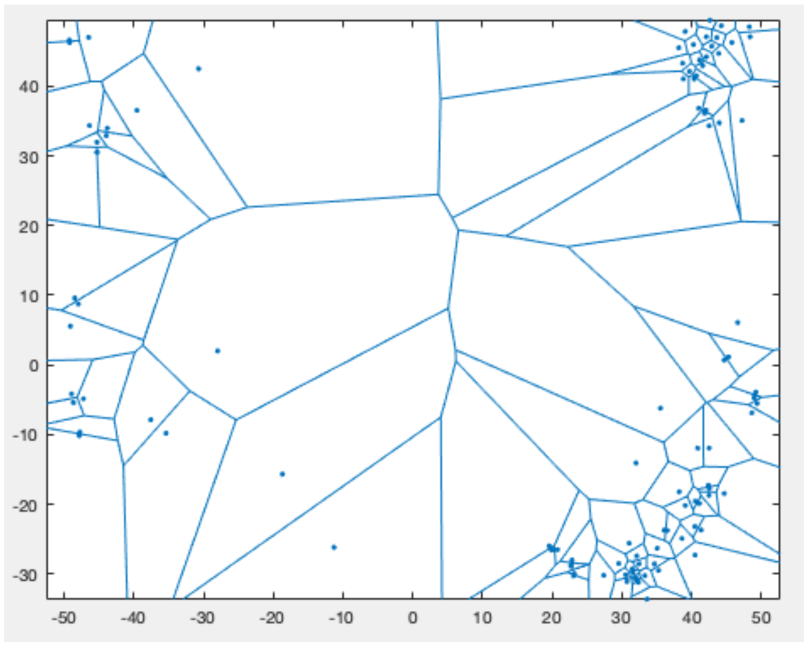

Figure 42: *Plot of configuration after 10000 iterations with KNN for k=8*

References

Argun, A., Callegari, A., & Volpe, G. (2021). *Simulation of complex systems* (1st ed.). IOP Publishing.

```
%% Standard Vicsek model.
close all
clear all
clc
tic
% PARAMETERS
t_steps = 10000;
N = 100;
L = 100;
R = 1;
dt = 1;
v = 1;
eta = 0.1;
limit = 45; % sight limit in degrees
k = 8;
%% EXERCISE 8.4 - 8.7 & 8.10 - 8.11
for m = 1:1
  [x, theta] = Initialize(N, L);
  gA = zeros(1, t_steps);
  gC = zeros(1, t_steps);
  for i = 2:t_steps
  neighbours = zeros(N) ~= 0;
  for j = 1:N
    neighbours(j, :) = Find(x, j, R, L);
  end
  velocities = UpdateV(v, theta, N);
  x = UpdatePos(x, velocities, dt, L);
%     theta = UpdateOrient(theta, neighbours, eta, dt,N);
  theta = UpdateOrientKNN(theta, x, k, eta, dt, limit, N); %unmute to use KNN, dont forget to mute row above
  gA(i) = GlobalAlignment(N, velocities, v);
  gC(i) = GlobalClustering(x, L, R);
  if (rem(i, 1000) == 0)
    disp(i)
  end
  end
end
%
%% PLOT
% 8.4 a
voronoi(x(:,1), x(:,2))
axis equal
% 8.4 b
f2 = figure;
plot(gA)
hold on
plot(gC)
legend('?', 'C')
ylim([0,1.1])
xlabel('t')
ylabel('? & C')
toc
%% FUNCTIONS
%Initialize positions and theta
function [pos,the] = Initialize(N, L)
   pos = zeros(N, 2);
   the = zeros(N, 1);
   for i = 1:N
    r1 = rand();
    r2 = rand();
    x = L*r1 - L/2;
    y = L*r2 - L/2;
    pos(i, :) = [x, y];
```

```matlab
        the(i) = rand()*2*pi;
     end
end

% Update positions with periodic boundary conditions
function pos = UpdatePos(pos, velocities,dt, L)
    for i = 1:size(pos, 1)
        dR = velocities.*dt;
        pos(i, :) = pos(i, :) + dR(i, :);
    end
    for i = 1:size(pos, 1)
        for j = 1:size(pos, 2)
            if (pos(i, j) > L/2)
                pos(i, j) = pos(i, j) - L;
            elseif (pos(i ,j) < -L/2)
                pos(i, j) = pos(i, j) + L;
            end
        end
    end
end

% Update velocities
function velocities = UpdateV(v, theta,N)
    velocities = zeros(N, 2);
    for i = 1:N
        velocities(i, 1) = v*(cos(theta(i)));
        velocities(i, 2) = v*(sin(theta(i)));
    end
end

% Find neighbors within Rf to update theta
function Neighbours = Find(p, index, Rf, L)
    Neighbours = zeros(1, size(p, 1)) ~= 0;
    current_p = p(index, :);
    % For cases when the absolute(dist) > L/2 = wrapped around
    for i = 1:size(p, 1)
        xdist = current_p(1) - p(i, 1);
        if (abs(xdist) > L/2)
            xdist = L - xdist;
        end
        ydist = current_p(2) - p(i, 2);
        if (abs(ydist) > L/2)
            ydist = L - ydist;
        end
        dist = sqrt(xdist^2 + ydist^2); %Euclidean
        if (dist < Rf)
            Neighbours(i) = true;
        end
    end
end
% Update theta
function theta = UpdateOrient(old_theta, neighbour, eta, dt,N)
    theta = zeros(N, 1);
    for i = 1:N
        r = rand - 0.5;
        S = find(neighbour(i,:));
        if (length(S) == 1)
            mean_theta = old_theta(i);
        else
            mean_theta = atan2(mean(sin(old_theta(S))), mean(cos(old_theta(S))));
        end
        theta(i) = mean_theta + eta*r*dt;
    end
end
% Find neighbors with KNN
function theta = UpdateOrientKNN(old_theta, x, k, eta, dt, limit, N)
    theta = zeros(N, 1);
    for i = 1:N
        dist = zeros(N, 1);
        for m = 1:N
            dist(m) = Inf;
        end
        for j = 1:N
            angle = acosd(dot(x(i, :), x(j, :))/(norm(x(i, :))*norm(x(j, :))));%cosine similarity
            if (angle < limit)
                dist(j) = norm(x(i, :) - x(j, :));
            end
        end
        [~, idx] = sort(dist);
        nearest = idx(1:k);
        mean_theta = atan2(mean(sin(old_theta(nearest))), mean(cos(old_theta(nearest))));
        theta(i) = mean_theta + eta*(rand - 0.5)*dt;
    end
end
% Calculate global alignment coefficient
function gA = GlobalAlignment(N, velocities, v)
    gA = norm(sum(velocities)) /(v* N);
end


%Calculate global clustering coefficient
function gA = GlobalClustering(x, L, Rf)
pos = [[x(:, 1) - L, x(:, 2) + L]; [x(:, 1), x(:, 2) + L]; [x(:, 1) + L, x(:, 2) + L];
     [x(:, 1) - L, x(:, 2)];     [x(:, 1), x(:, 2)];     [x(:, 1) + L, x(:, 2)];
     [x(:, 1) - L, x(:, 2) - L]; [x(:, 1), x(:, 2) - L]; [x(:, 1) + L, x(:, 2) - L]];
[vertices, cells] = voronoin(pos);
area_idx = zeros(1, size(pos, 1));
area = zeros(sum(area_idx), 1);

for i = 1:size(pos, 1)
    if (max(abs(pos(i, 1))) < 1/2*L)
        if (max(abs(pos(i, 2))) < 1/2*L)
            area_idx(i) = 1;
        end
    end
end

j = 1;
for i = find(area_idx)
    V1 = vertices(cells{i}, 1);
    V2 = vertices(cells{i}, 2);
    area(j) = polyarea(V1, V2);
    j = j + 1;
end
gA = sum(area < pi*Rf^2)/length(area);
end
```