5-24-2024

Investigating the Connectedness of a Gorbonos Insect Swarm

Brendan Perez
Linfield University

Investigating the Connectedness of a Gorbonos Insect Swarm

Brendan Perez

A THESIS

Presented to the Department of Physics
LINFIELD COLLEGE
McMinnville, Oregon

In partial fulfillment of the requirements
for the Degree of

BACHELOR OF SCIENCE

May, 2024

Thesis Acceptance

Linfield College

Thesis Title: Investigating the Connectedness of a Gorbonos Insect Swarm

Submitted by: Brendan Perez

Date Submitted: May, 2024

Thesis Advisor: _____

Dr. Joelle Murray

Physics Department: _____

Dr. Michael Crosser

Mathematics Department: _____

Dr. Kate Lorenzen

ABSTRACT

Investigating the Connectedness of a Gorbonos Insect Swarm

Insect swarms exhibit collective behaviors that emerge from the interactions between individual insects. In midges, these interactions are thought to be governed by long-range acoustic signals from other insects in the swarm. A model developed by Gorbonos *et al.* [1], [2] adds the long-range acoustic behavior into an equation of motion to describe the midge swarm dynamics. This thesis expands on Gorbonos' model by considering various subsets of insect-insect interactions, and investigates how diffusion and polarization parameters vary with the number of interactions.

Chapter 1

Introduction

In many complex systems, there are instances where simple individual behavior leads to large-scale emergent behavior. In biology, we can see this in animal behavior. Birds can be characterized by flocking, in which individual birds move together as a group in the same direction. Fish can characterized by milling when attacked by a predator, where individuals rotate together in a ball to act defensively. Examples can also be seen in physics, such as an array of iron atoms at a sufficiently cold temperature leading to a large-scale alignment of spin.

In this paper, we investigate the behavior of swarming found in insects. Here, individual behavior seems random but as with the previous examples, emergent behavior arises in the collective swarm of insects. Several models have already been thoroughly investigated, including the Viscek model [3], the random walk model, and the Gorbonos model [1], [2]. In the Viscek model, insects are given random positions and velocities under certain parameters. Then, each insect finds all other insects less than a given distance, and moves in the average direction of those insects. A random perturbation may be added to the new direction. This model is illustrated in Fig. 1.1.

The random walk builds upon the Viscek model, now extending it to three di-

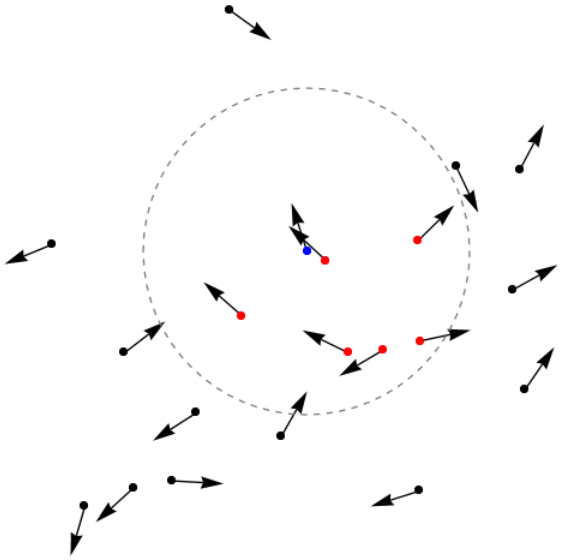

Figure 1.1: Figure illustrating the Viscek model. The target insect is in blue, insects within range are in red, and insects out of range are in black. Only insects within the circle contribute to the target insect's new direction, illustrated by the arrows.

mensions. An extra weighting is given to insects moving along the z-axis [1]. The Gorbonos model will be the focus of this thesis. Unlike the previous two models, this model is completely deterministic. That is, there is no effect due to randomness. Gorbonos' model treats the insects as identical point masses acting as if there was a force, mathematically analogous to universal gravitation. However, unlike universal gravitation, interactions are tuned out when insects are sufficiently close together. We can choose when to tune out these interactions using an adaptive radius, denoted by R_{ad}.

Out of all three of these models, computing Gorbonos' model is by far the most computationally expensive. Since every insect interacts with all but one, the total number of interaction grows roughly with the square of the number of insects. However, we only need to compute half of these since interactions are symmetric. For example, with 100 insects, a total of $\frac{1}{2} \cdot 100 \cdot 99 = 5050$ interactions are used. Using Gorbonos' model, many of these interactions contribute nothing to the overall effect

of the swarm, so it is reasonable to assume that it is not necessary to compute every interaction. Furthermore, it is also not reasonable that every insect actually interacts with every other one.

We now introduce the Laplacian model. In this model, we only consider a given subset of interactions found between insects. We can represent such interactions using graphs, where vertices represent the insects and edges represent the interactions present between insects. For example, given insects $\{1, 2, 3, 4, 5\}$ and interactions $\{1 \leftrightarrow 2, 1 \leftrightarrow 3, 1 \leftrightarrow 5, 2 \leftrightarrow 4, 3 \leftrightarrow 4, 3 \leftrightarrow 5\}$, we illustrate the corresponding graph found in Fig. 1.2. The lines in this figure represent the interactions that we have selected between these insects.

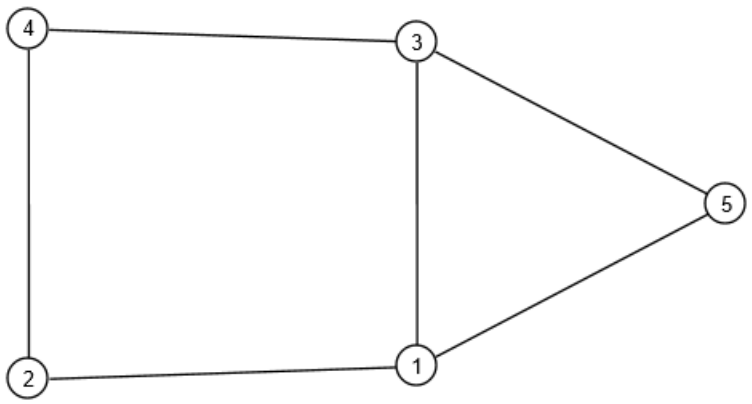

Figure 1.2: Example insect swarm with interactions given by $\{1 \leftrightarrow 2, 1 \leftrightarrow 3, 1 \leftrightarrow 5, 2 \leftrightarrow 4, 3 \leftrightarrow 4, 3 \leftrightarrow 5\}$.

Furthermore, we define the neighborhood of an insect i as the set of insects it interacts with, given by $N(i)$. Using the previous example, we have $N(1) = \{2, 3, 5\}$, $N(2) = \{1, 4\}$, $N(3) = \{1, 4, 5\}$, $N(4) = \{2, 3\}$, and $N(5) = \{1, 3\}$.

Using the Laplacian model, we introduce two further models [4]. The minimum spanning tree (MST) model represents the bare minimum a swarm can be connected. Out of all subsets of interactions that connect a swarm together, the MST model is the

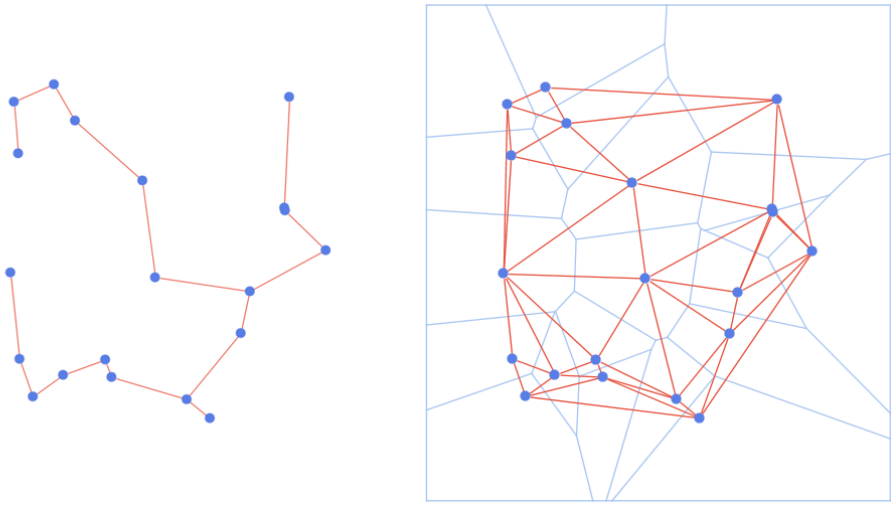

Figure 1.3: Graph illustrating the interactions present in the MST and DT models for a randomly generated set of 20 insects. The same insect swarm is used. On the left, the MST model is shown, with the minimum distance of interactions needed to have a swarm connected. On the right, the DT is model is shown. The Voronoi cells are shown about each insect, and insects with adjacent Voronoi cells are connected.

one with the shortest total distance between interactions. An example is illustrated in Fig. 1.3 (left). The next model is the Delaunay Triangulation (DT) model. In this model, we divide space into regions whose nearest insect is fixed, called *Voronoi cells*. Then, an interaction is present between two insects only if their Voronoi cells are adjacent. These regions and interactions are illustrated in Fig. 1.3 (right). This model should effectively capture only the nearest neighbors found in an insect swarm.

In two dimensions, we see that the lines dividing the Voronoi cells are perpendicular bisectors between pairs of insects, and the points bounding three regions are the circumcenters of some three insects. Finally, we compare these two models to the original Gorbonos model, using a complete set of interactions. This is not illustrated since there would be a line connecting every pair of insects. Using the previous example swarm, we see that the MST model uses 19 interactions, the DT model uses 49 interactions, and the original Gorbonos model uses 190 interactions.

Using these three Gorbonos-based models, we now have a series of swarms whose

number of interactions varies. In this paper, we show that the MST model is too sparsely connected to model a swarm, whereas the DT model is able to effectively capture the same results as the full Gorbonos swarm.

Chapter 2

Theory

In Gorbonos' paper, the key equation which describes the insect-insect interactions in a swarm is

$$\mathbf{F}_i = C \sum_{j=1}^{n} \frac{\hat{\mathbf{r}}_{ij}}{|\mathbf{r}_i - \mathbf{r}_j|^2} \frac{R_{ad}^{-2}}{R_{ad}^{-2} + \sum_{k=1}^{n} |\mathbf{r}_i - \mathbf{r}_k|^{-2}}, \tag{2.1}$$

where the effective force $\mathbf{F}_i$ represents the net interactions acting on insect i, whose positions are given by $\mathbf{r}_i$. Like Newton's law of universal gravitation, there is a constant C that describes the universal attraction between insects, and a term $\hat{\mathbf{r}}_{ij}/|\mathbf{r}_i - \mathbf{r}_j|^2$ corresponding to the inverse square of distance between insects i and j. Finally, an additional term is added in order to tune out interactions within an adaptive radius R_{ad}. Using Newton's second law, we write $\mathbf{F}_i = m_i \mathbf{r}_i''(t)$, where $\mathbf{r}_i''(t)$ is the acceleration of insect i. We can further assume all insects have identical mass, say m, and can absorb this constant into C, calling it C_1. We then have the following equation:

$$\mathbf{r}_i''(t) = C_1 \sum_{j=1}^{n} \frac{\hat{\mathbf{r}}_{ij}}{|\mathbf{r}_i - \mathbf{r}_j|^2} \frac{R_{ad}^{-2}}{R_{ad}^{-2} + \sum_{k=1}^{n} |\mathbf{r}_i - \mathbf{r}_k|^{-2}} \tag{2.2}$$

For the Laplacian model, the sums in this equation now go over the insects found in $N(i)$ for insect i. That is,

$$\mathbf{r}_i''(t) = C_1 \sum_{j \in N(i)} \frac{\hat{\mathbf{r}}_{ij}}{|\mathbf{r}_i - \mathbf{r}_j|^2} \frac{R_{ad}^{-2}}{R_{ad}^{-2} + \sum_{k \in N(i)} |\mathbf{r}_i - \mathbf{r}_k|^{-2}} \tag{2.3}$$

We now have a second-order differential equation, which we now solve numerically. The most basic approach is to use Euler's method [5]. Given differential equation $\mathbf{x}'(t) = f(\mathbf{x}, t)$ and initial condition $\mathbf{x}(0) = \mathbf{x}_0$, current position $\mathbf{x}(t)$, next position $\mathbf{x}(t + dt)$, and time step dt, we have

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, t) \tag{2.4}$$

$$\Rightarrow \frac{\mathbf{x}(t + dt) - \mathbf{x}(t)}{dt} \simeq f(\mathbf{x}, t) \tag{2.5}$$

$$\Rightarrow \mathbf{x}(t + dt) - \mathbf{x}(t) \simeq f(\mathbf{x}, t)dt \tag{2.6}$$

$$\Rightarrow \mathbf{x}(t + dt) \simeq \mathbf{x}(t) + f(\mathbf{x}, t)dt. \tag{2.7}$$

From this equation, we can visualize Euler's method as given current point x_i, the next point will change in the direction of the derivative. As dt goes to zero, we converge upon our solution for a given initial condition. Despite this, Euler's method does not work well for many systems like planetary motion. For example, in a system where a planet orbits around the sun, it can be shown that no matter how small dt is, the planet always spirals away from the sun eventually, although it takes longer to do so with smaller dt. These are illustrated for the following system of equations,

using $dt = 0.5$ and $dt = 0.1$ respectively in Fig. 2.1. The curved, blue arrows indicate the direction of the derivative and the bold, black arrow indicates the solution from Euler's method. It is not hard to verify the actual solution is a closed circle given by $(r\cos(t), r\sin(t))$ for $\mathbf{x}(0) = (r, 0)$.

$$\begin{cases} x'(t) = -y(t), \\ y'(t) = x(t) \end{cases} \tag{2.8}$$

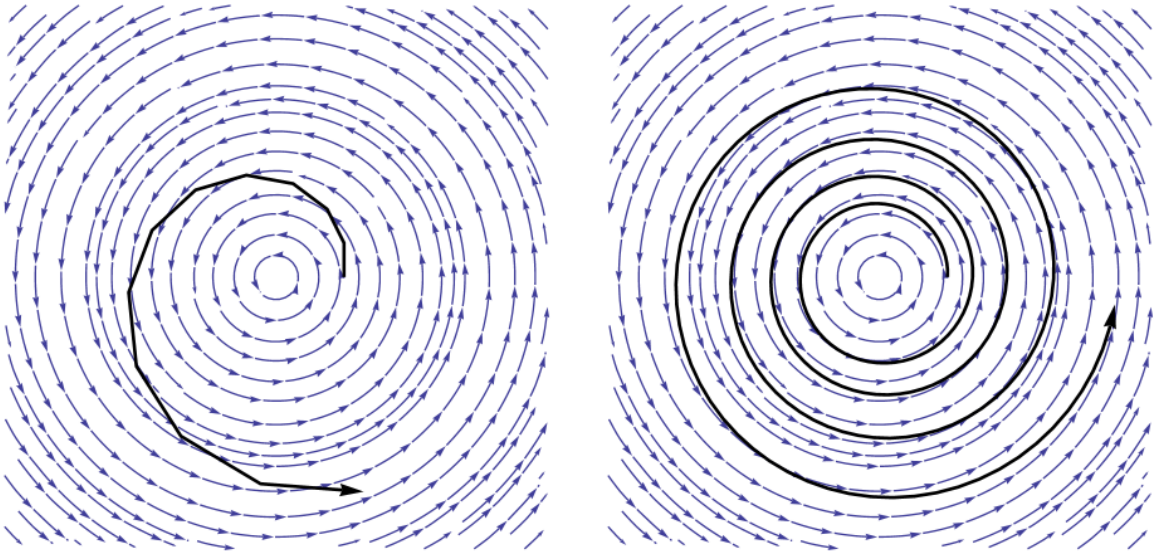

Figure 2.1: Euler's method for the above system with $\mathbf{x}(0) = (0.25, 0)$ and $dt = 0.5$ (left) and $dt = 0.1$ (right).

To solve the problems found in Euler's method, we use the fourth-order Runge-Kutta method, chosen for its robust solution. As before, our system is given by differential equation $\mathbf{x}'(t) = f(\mathbf{x}, t)$ with initial condition $\mathbf{x}(0) = \mathbf{x}_0$, and we use the following iterative process. Given time step dt, we define $\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$ as follows:

$$\mathbf{k}_1 = f(\mathbf{x}(t), t) \tag{2.9}$$

$$\mathbf{k}_2 = f(\mathbf{x}(t) + \frac{1}{2}\mathbf{k}_1, t + \frac{1}{2}dt) \tag{2.10}$$

$$\mathbf{k}_3 = f(\mathbf{x}(t) + \frac{1}{2}\mathbf{k}_2, t + \frac{1}{2}dt) \tag{2.11}$$

$$\mathbf{k}_4 = f(\mathbf{x}(t) + \mathbf{k}_3, t + dt) \tag{2.12}$$

Next, we compute $\mathbf{x}(t+dt) = \mathbf{x}(t) + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$. Furthermore, we decompose a system of second-order differential equation of n variables and n equations into a system of first-order of $2n$ variables and $2n$ equations. The added variables are $\mathbf{v}(t) = \mathbf{x}'(t)$ and the added equations are $\mathbf{v}'(t) = \mathbf{x}''(t)$. Since our model is in three dimensions, given n midges we end up solving a system of $6n$ first-order differential equations. In the Laplacian models, the sums found in Gorbonos' equation now only run over insects that are connected, significantly reducing the computation time.

2.0.1 Diffusiona and Polarization

Given an insect swarm with given positions and velocities, we now define the relevant parameters necessary to analyze swarming data. Firstly, we define the center of mass of a swarm to be the average position vector of the insects, given by $\langle \mathbf{r} \rangle$. The angled brackets are used to denote the average over all insect positions $\mathbf{r}_i$. Then, diffusion is defined to be the average value of the distance squared from every insect to the center of mass, given by

$$\langle R^2 \rangle = \frac{1}{n} \sum_{i=1}^{n} |\mathbf{r}_i - \langle \mathbf{r} \rangle|^2. \tag{2.13}$$

The choice for diffusion as a parameter becomes relevant when considering random

walks in general. For example, given a one-dimensional random walk, with all initial positions at zero, and a $50-50$ chance of increasing or decreasing by one, it can be shown that diffusion increases linearly with time. We need diffusion to be centered around the center of mass since otherwise diffusion would increase quadratically over time. This parameter is necessary to compare the Gorbonos models to the other models, including the Viscek and the random walk models.

Another parameter of interest is to see how far the center of mass drifts away from its starting point, given by $|\langle \mathbf{r} \rangle|$. We would expect this to increase linearly over time on average. This can be seen using Newton's law of gravitation, where even though individual bodies accelerate, the center of mass has no acceleration and therefore constant velocity. We call this velocity the swarm velocity, and can find this by finding the slope of the best fit line.

Next, we define the polarization Φ [6] as

$$\Phi = \left| \frac{1}{n} \sum_{i=1}^{n} \hat{\mathbf{v}}_i \right|. \tag{2.14}$$

This gives a measurement of how aligned an insect swarm is, giving us a number between 0 and 1, done by averaging the unit velocities of each insect and taking the magnitude. For flocking birds, high polarization values have been observed, typically over 0.9. For swarming behavior, much lower values have been observed, usually between 0.15 and 0.30. If all insects are in the same direction, the polarization is one. If there is a 50-50 split with insects moving in opposite directions, polarization is zero. For two insects moving in random directions, the expected polarization is $2/\pi = 0.6366....$ By computing these parameters over time, we can then collect data for 100 simulated swarms and analyze this data.

Chapter 3

Methods

Now, we illustrate how the swarm data is actually computed, done in Python. First, the initial insect positions and velocities are initialized over a swarm of n insects. The positions are chosen uniformly in a sphere of radius R_0 centered about the origin, and the velocities are chosen from a standard normal distribution in the x, y, and z directions. This data is stored as a $n \times 6$ array, with a row for each insect i, three columns for position, and three columns for velocity.

For each insect i, we store its neighbors $N(i)$ as a list. Using all n lists, we create a list of lists to store every interaction. Using the example from figure 1.2, we have our list of lists as

$$[[2, 3, 5], [1, 4], [1, 4, 5], [2, 3], [1, 3]]. \tag{3.1}$$

Storing the interactions this way allows the effective force to be computed efficiently, as once the insect graph is determined, these interactions are stored to memory. For the original Gorbonos model on n insects, our list of lists has the form:

$$[[2, 3, 4..., n-1, n],$$

$$[1, 3, 4, ..., n-1, n],$$

$$\vdots$$

$$[1, 2, 3, ..., i-1, i+1, ..., n-1, n],$$

$$\vdots$$

$$[1, 2, 3, ..., n-2, n],$$

$$[1, 2, 3, ..., n-2, n-1]].$$

Next, we show how to compute these interactions for the MST and DT models.

3.1 Minimum Spanning Tree

To compute the minimum spanning tree of a set of insects weighted by distance, we use Kruskal's algorithm. First, all distances are computed between every pair of distances. Then, distances are sorted from smallest to largest. Initially, no interactions are selected. Then, starting with the pair of insects with smallest distance, we add these interactions to our set if a cycle is not formed until all edges are gone through. That is, there is no sequence of interacting insects $i_1, i_2, ..., i_k$ has i_1 interacting with i_k for $k \geq 3$.

Consider the swarm found in figure 3.1. The distances between pairs of insects are in Table 3.1. The smallest distance is 0.246438, found between insects 3 and 5, so we add this interaction. The next smallest distance is 0.262878, found between insects 1 and 3, so we add this interaction. We keep adding edges until we reach 0.445604, found between insects 1 and 5. However, this forms a cycle between insects

1, 3, and 5, so we do not include this interaction. After going through the rest of the interactions, we obtain the MST graph for this swarm, which is highlighted in red in figure 3.1.

	1	2	3	4	5
1	0	0.732796	0.262878	0.77002	0.445604
2	0.732796	0	0.867282	0.394099	0.819121
3	0.262878	0.867282	0	0.778288	0.246438
4	0.77002	0.394099	0.778288	0	0.623367
5	0.445604	0.819121	0.246438	0.623367	0

Table 3.1: Table illustrating distances between insects found in figure 3.1. Given row i and column j, the ij-entry indicates the distance between insects i and j.

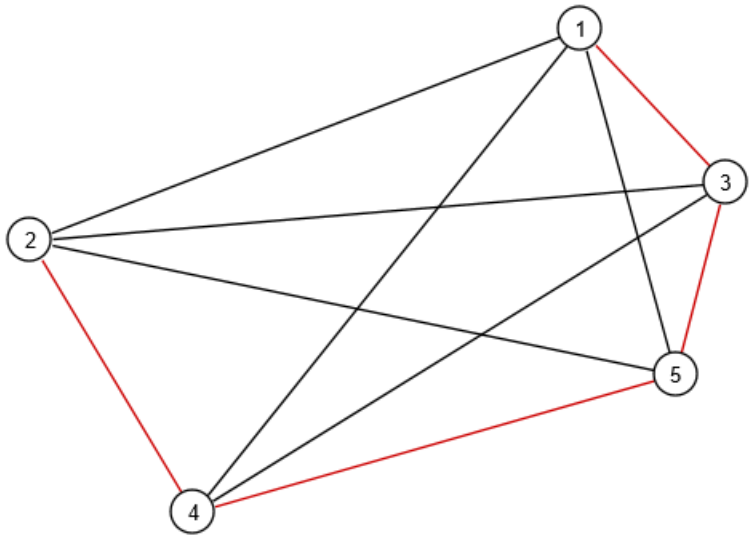

Figure 3.1: Example illustrating how to construct the MST graph for the Laplacian model.

3.2 Deluanay Triangulation

To determine the Delaunay triangulation of a swarm, we first find regions in space whose nearest insect is the same. In practicality, it is much easier to find the boundaries of these regions, whose distances are the same between two or more insects. The lines bounding these regions are necessarily the perpendicular bisectors of

some pair of points, and the points where lines intersect are necessarily the circumcenters for the triangles formed in the triangulation. The circumcenter of a triangle is the unique point whose distance from all three points is the same. The circle centered at the circumcenter that goes through these points is the circumcircle.

In three dimensions, an analogous result holds. Planes represent regions that are equidistant from two or more points. Lines bounded by two planes are regions that are equidistant to three or more nearest neighbors. These lines are drawn perpendicular to the plane formed from these three points, going through the circumcenter of this triangle. Points bounded by three planes are regions that are equidistant to four or more nearest neighbors. Such points are the circumcenter of the tetrahedra formed by these points, and the sphere centered at the circumcenter that goes through these four points is the circumsphere.

Knowing this, we now illustrate the Bowyer-Watson algorithm for two dimensions. For three dimensions, tetrahedra are used instead of triangles, and circumspheres are used instead of circumcircles. Rather than keeping track of edges, triangles, or unordered triples of insects are kept track of. First, an auxillary set of three points (or four for three dimensions) is created so that all points are enclosed in the triangle formed by these points. Points are then added one at a time. If any triangle formed by the previous set of points has its circumcircle enclose the new point, that triangle is removed and replaced with triangles connected to the added point. After all points are added, any edges connecting to the auxillary triangle are removed. An example illustrating this process is shown in Fig. 3.2 and Fig. 3.3.

A more intuitive way to visualize how to form the interactions in this swarm is to center circles of fixed radius around each insect, so that no two circles overlap.

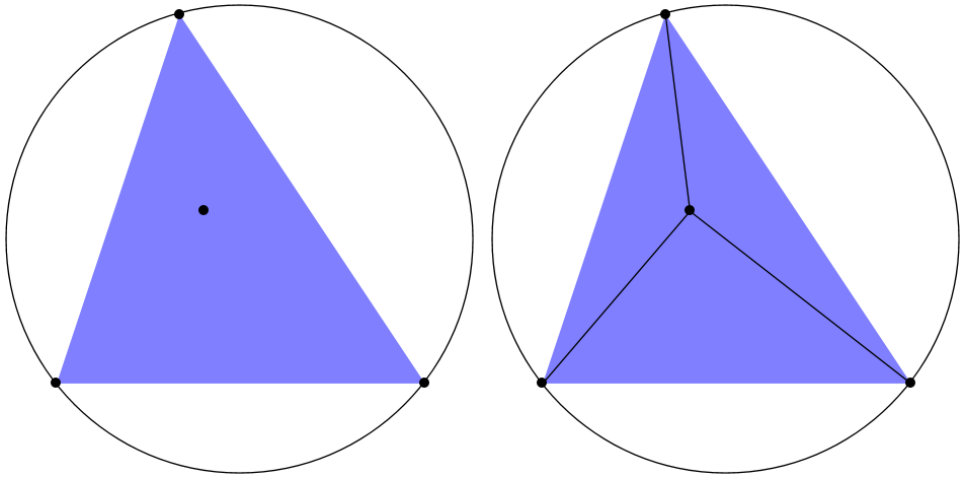

Figure 3.2: First case of using the Bowyer-Watson algorithm. If the added point is within some triangle, the triangle corresponding to that circumcircle is replaced with three triangles connected to the added point.

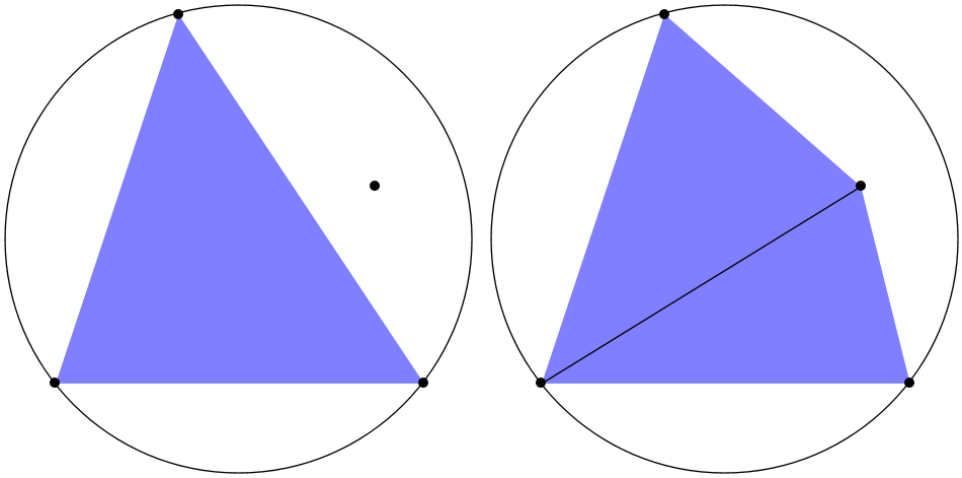

Figure 3.3: Second case of using the Bowyer-Watson algorithm. If the added point is outside a triangle but within its circumcircle, the triangle corresponding to that circumcircle is replaced with two triangles connected to the added point.

Then, grow all circles outwards at the same rate. When two circles meet, they cluster together like bubbles, forming a linear boundary between the two. An interaction between the corresponding insects is then formed. However, this does not provide a practical way to actually compute the Delaunay triangulation of a swarm.

3.3 Diffusion and Polarization

To compute these parameters of interest, we store separate lists containing these variables over time. For diffusion, we compute the center of mass at each time step for the swarm, and subtract these values from every insect's position. We then take the average value of distance squared of this difference, giving us our diffusion at this time. To compute polarization, we normalize every insect's velocity to have magnitude one. Then, we take the average of the normalized velocities and find its magnitude. This gives our polarization.

For diffusion, we expect this to increase linearly over time, much like the random walk model. Even though this is a deterministic model, this system is chaotic enough to the point that it appears to be stochastic. For polarization, we expect to obtain values in the 0.15 to 0.30 range as found from previous results [6]. To analyze data for these parameters, we run this model 100 times for a given period of time. Then, at each time step, we obtain a distribution of diffusions and polarizations, aiming to compare these values to known results. We then plot the mean of the 100 runs, as well as one standard deviation away from the mean for both diffusion and polarization. We also plot the distance of the center of mas from the origin as a reference point.

Chapter 4

Results

When running the Laplacian model for each of the MST, DT, and complete graphs, the following parameters were used:

- $n = 30$ insects were used
- A timestep of $dt = 0.1$ was used
- 100 runs were used
- An adaptive radius of $R_{ad} = 1$ was used
- The swarm was run until $t = 10$, giving a total of $10/0.1 = 100$ iterations
- The initial swarm was chosen uniformly in a sphere of radius $R_0 = 1$
- The initial swarm had velocity chosen from a standard normal distribution

The only difference between these models was the choice of C, since by reducing the number of interactions, the total force per insect was significantly reduced. The following choices were made:

- For the Minimum Spanning Tree model, we will show that no choice of C works
- For the Delaunay Triangulation and Complete Gorbonos model, $C = 5$

4.1 Minimum Spanning Tree Model

We first investigate why no choice of C allows a swarm to be connected. I investigated three different choices of C: 1, 5, and 10. When $C > 10$, the numerical method fails to work with fixed $dt = 0.1$, as the changes in insect positions become too large. The diffusion plots for each case is shown in Fig. 4.1. It can be seen that in all cases, diffusion increases quadratically over time, indicating the swarm is continually expanding apart. Hence, the MST model has too few connections for a stable insect swarm. Note that the rate of growth decreases from $C = 1$ to $C = 5$, but increases from $C = 5$ to $C = 10$. This indicates that $C = 10$ is too large of a universal attraction between insects, as the changes in positions become too random.

Interestingly, the polarization seems to be stable over time and is in the correct range, shown in in Fig. 4.2. However, as polarization is solely determined by velocity and not position, this has no bearing on the fact that the insect swarm is too spread out. As C increases, note that the roughness of the polarization plot increases, indicating more sudden changes in velocities. Also, in the plot for $C = 10$, the polarization takes longer to reach its maximum, indicating that the choice of C is also too large.

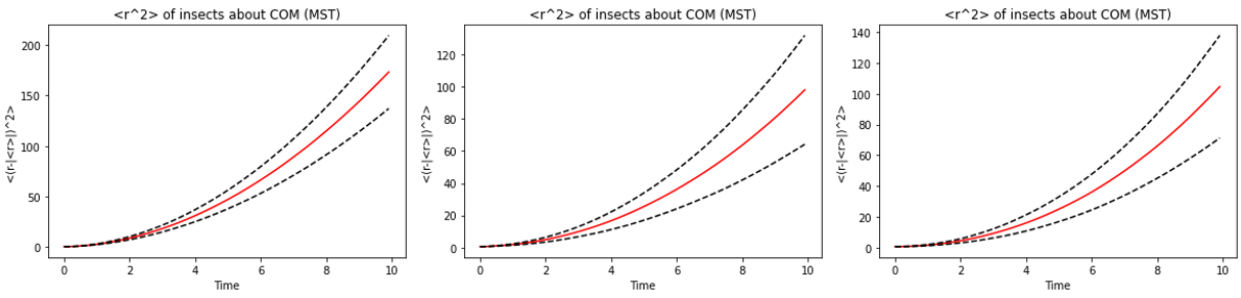

Figure 4.1: Diffusion $\langle R^2 \rangle$ of the MST model. From left to right, the choices for C are 1, 5, and 10 respectively. The solid red line is the mean over 100 runs, and the dashed black lines are one standard deviation away from the mean.

These results are reasonable since in all three cases, the swarm velocity was within

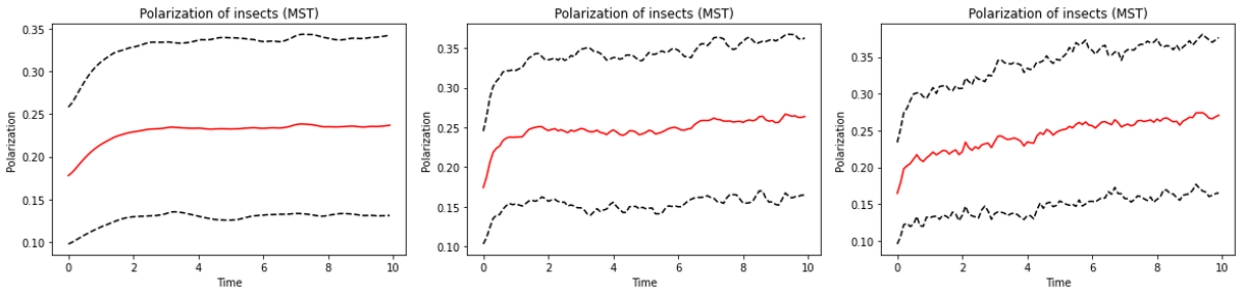

Figure 4.2: Plots showing the polarization of the MST model. From left to right, the choices for C are 1, 5, and 10 respectively. The solid red line is the mean over 100 runs, and the dashed black lines are one standard deviation away from the mean.

$0.385 - 0.395$ for each model (using time and distance units from this model), found using the best fit line through all points except the first ten. The choice for this exclusion is made clear in Fig. 4.3.

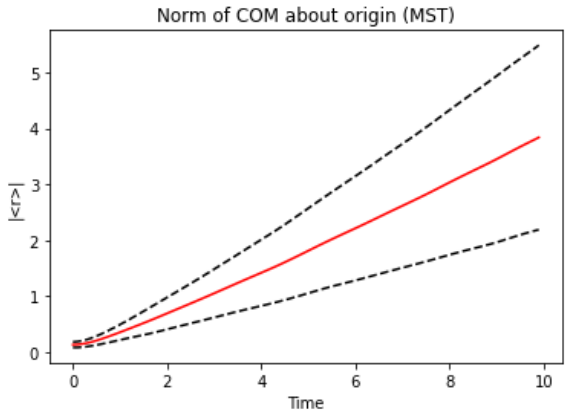

Figure 4.3: Figure illustrating why the first few points were excluded to calculate swarm velocity. The solid red line is the mean over 100 runs, and the dashed black lines are one standard deviation away from the mean.

4.2 Delaunay Triangulation Model

Now, we investigate the results form the Delaunay Triangulation model. Here, a choice of $C = 5$ was used. Unlike the MST model, the diffusion did not grow quadratically. Interestingly, as seen in Fig. 4.4, the diffusion seems to oscillate a bit before settling down to a linear growth rate. Using linear regression, this line has a

slope of 0.076, which is extremely low. This indicates that the Delaunay model is connected enough to hold a swarm of insects together.

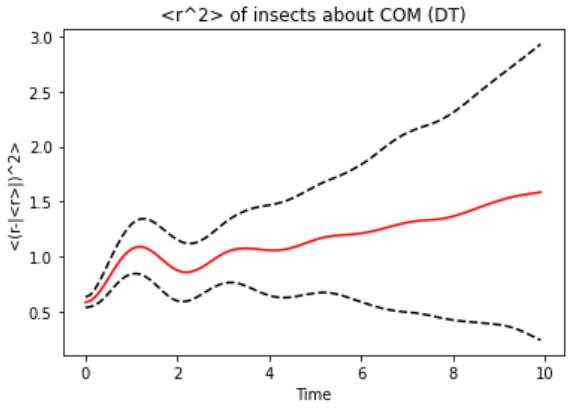

Figure 4.4: Diffusion for the DT model. Note that the diffusion growth is initially oscillatory, but settles down to a linear rate after 2 time units. The solid red line is the mean over 100 runs, and the dashed black lines are one standard deviation away from the mean.

Next, we examine the polarization of this model, shown in in Fig. 4.5. The polarization of this model is around 0.25, just like the MST model. This indicates that the polarization found in this model is around the same as insect swarms found in nature, which supports this model.

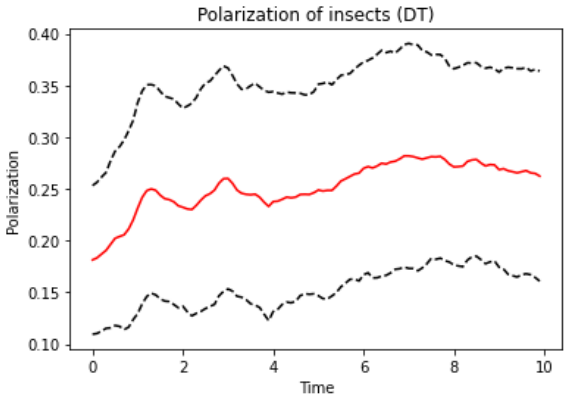

Figure 4.5: Polarization for the DT model. Like the MST model, polarization remains around 0.25. The solid red line is the mean over 100 runs, and the dashed black lines are one standard deviation away from the mean.

Interestingly, the swarm velocity found using the same method as the MST model,

was 0.386. This is right around the same rate as the MST model. This indicates that the assumption that the center of mass has no acceleration on average is likely correct.

4.3 Complete Gorbonos Model

Finally, we examine the complete Gorbonos model. By looking at the diffusion plot in Fig. 4.6, we see that like the Delaunay Triangulation model, the diffusion oscillates before settling down to a linear growth rate. Using linear regression, we find this rate to be 0.099, which is very close to the diffusion growth rate found in the Delaunay Triangulation model.

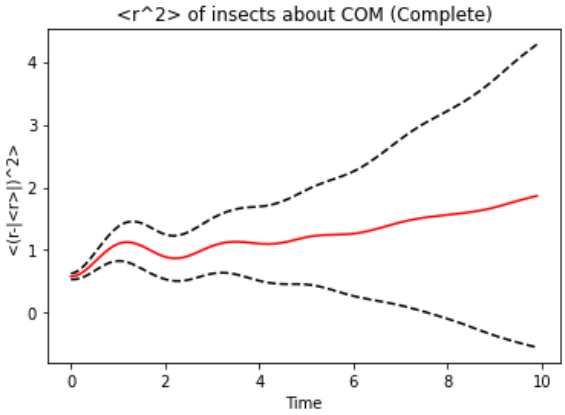

Figure 4.6: Diffusion for the Complete Gorbonos model. Like the DT model, the diffusion oscillates before settling down, and has a diffusion rate very similar to the one found in the DT model. The solid red line is the mean over 100 runs, and the dashed black lines are one standard deviation away from the mean.

We see that in Fig. 4.7, the polarization plot also appears to be very similar to both the MST and DT models. Again, a diffusion of around 0.25 is reached.

Finally, a swarm velocity of 0.368 was found in this model. While noticeably lower than the other two models, this rate is still very close to the MST and DT models.

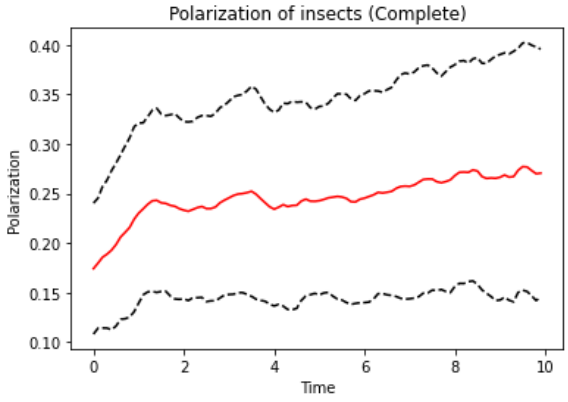

Figure 4.7: Polarization for the Complete Gorbonos model. Like the other two models, a polarization of around 0.25 is reached. The solid red line is the mean over 100 runs, and the dashed black lines are one standard deviation away from the mean.

4.4 Conclusion

From these results, I have found that the MST model is too sparsely connected to be considered a swarm. By looking at the diffusion rates for various choices of C, all of them lead to a quadratic growth rate. This indicates that insects are acting almost independently of each other, and are therefore not swarming. Even though the polarization may be correct for an insect swarm, this is still not sufficient to determine swarming behavior.

Next, I have found that the Delaunay Triangulation model accurately represents the Complete Gorbonos model in both diffusion, polarization, and swarm velocity. Both diffusion rates are on the order of 0.1, which indicates the swarm is coherent enough to hold together for extended periods of time. Knowing that diffusion is the average distance to the center of the swarm squared, the average distance to the center is even smaller, indicating a very strongly connected swarm. While one variable is not usually enough to conclude that two models are sufficiently similar, having these models agree in all three parameters strongly suggest that the Delaunay Triangulation model is an accurate representation of the Complete Gorbonos model.

The relevant results are shown in Table 4.1, using $C = 5$ for the MST model.

Model	Swarm Velocity	Swarm Diffusion	Polarization
MST	0.387	N/A	0.248
DT	0.386	0.076	0.253
Complete	0.368	0.099	0.246

Table 4.1: Main results for each of the three models. Both swarm velocity and polarization were consistent for each model. However, the diffusion rate of the MST model was not linear, and therefore does not have a value in this table.

Chapter 5

Future Directions

Given that the MST model was too sparsely connected and that the Delaunay Triangulation model was sufficiently connected, the natural question is to ask at what point is a swarm of n insects connected. To answer this, we now need to create graphs with some fixed number of m edges. One possible approach would be to randomly select the m edges from the $n(n-1)/2$ possible ones. However, this would not likely work as pairs of distant insects are equally likely to be selected as close insects.

As a result, one could weight each pair of insects by the reciprocal of distance squared, and randomly select the edges from this weighted distribution. This would ensure that neighboring insects would be more likely to be selected. However, another problem is presented. This method is very likely to form clusters of independent insects, since this would select all the interactions within a cluster first before joining the clusters together. Hence, the next step would be to find a way to choose any number of interactions, while ensuring a reasonably connected swarm.

Another future step would be to investigate the comparison between the Delaunay Triangulation model and the Complete Gorbonos model more thoroughly. For example, investigate how these two models compare for different values of C or R_{ad},

or even different initial swarms. Then, this model could be compared against the other swarming models described in this paper, such as the random walk and Viscek models.

Appendix A

Python Code

```
    # -*- coding: utf-8 -*-
"""
Created on Wed Jul  5 10:55:32 2023

@author: Brendan Perez
"""
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import Delaunay
from scipy.sparse import csr_matrix
from scipy.sparse.csgraph import minimum_spanning_tree

model = "Complete" #Complete, MST, or DT
n = 30 #number of midges
steps = 100 #number of iterations to run code for
runs = 100 #number of runs in multirun
l = 1 #initial radius of midges
R_ad = 1 #adaptive radius
R_r = 0.5 #repulsion radius
C = 2 #universal attraction constant
dt = 0.1 #time step for RK4 method
vicsek = 0 #perturbation of velocity in x, y, z, directions
animate_on = False
com_fixed = False #plots flies with com at origin
diffusion_plot = False
polarization_plot = False
repulsion_on = False #adds repulsion factor from equation 41
multirun_on = True #turns on multirun feature
def initialize(n,l):
    r = [] #initializing array of positions
    while len(r)<n: #uniform distribution in sphere of radius l
        x = 2*l*np.random.rand(3)-np.array([l,l,l])
        if np.linalg.norm(x)<l:
```

```
            r.append(list(x))
    r = np.array(r)
    v = np.random.normal(loc=0,scale=1,size=(n,3))
    return np.concatenate((r,v),axis=1)
x = initialize(n,l)

def pairwise(pts):
    return csr_matrix([[np.linalg.norm(pts[i,:]-pts[j,:])
    for i in range(len(pts))] for j in range(len(pts))])

def insect_mst(pts):
    A = pairwise(pts)
    x = csr_matrix.toarray(minimum_spanning_tree(A))
    x+=x.T
    return [list(np.nonzero(x[i])[0]) for i in range(n)]

def insect_delaunay(pts):
    edges = Delaunay(pts).simplices
    x = np.zeros([n,n],dtype=int)
    for i in edges:
        for j in range(len(i)):
            for k in range(j):
                x[i[j],i[k]]+=1
    x+=x.T
    return [list(np.nonzero(x[i])[0]) for i in range(n)]

if model=="MST":
    interactions = insect_mst(x[:,:3])
elif model=="DT":
    interactions = insect_delaunay(x[:,:3])
else:
    interactions = [list(filter(lambda x: x!=i,range(n)))
    for i in range(n)]

def f1(r,i): #gives first sum in equation 1 in Gorbonos et al with index j
    if repulsion_on:
        return sum((r[j]-r[i])/np.linalg.norm(r[j]-r[i])**3*
        (1-2*np.exp(-np.linalg.norm(r[j]-r[i])**2/R_r**2))
        for j in filter(lambda x: x!=i, range(n)))
    if not repulsion_on:
        return sum((r[j]-r[i])/np.linalg.norm(r[j]-r[i])**3
        for j in interactions[i])
def f2(r,i): #gives second sum in equation 1 in Gorbonos et al with index k
    return sum(np.linalg.norm(r[k]-r[i])**-2 for k in interactions[i])
def f(r,i): #gives the net acceleration on midge i
```

```
    return C*f1(r,i)*R_ad**-2/(R_ad**-2+f2(r,i))
def g(x): #derivative of x = [r,v], with x" = [v, f(r,i)] for insect i
    v = np.random.normal(loc=0,scale=vicsek,size=(n,3))
    w = np.concatenate((v,np.zeros([n,3])),axis=1)
    w+=np.array([list(x[i,3:])+list(f(x[:,:3],i)) for i in range(n)])

def update(x): #updates positions/velocities with RK4 method
    k0 = dt*g(x)
    k1 = dt*g(x+k0/2)
    k2 = dt*g(x+k1/2)
    k3 = dt*g(x+k2)
    return x+(k0+2*k1+2*k2+k3)/6

diffusion = [] #time, <r^2> value of insects about COM, |<r^2>|
polarization = []
if animate_on or diffusion_plot or polarization_plot:
    for i in range(steps):
        print(i)
        COM = np.array([np.average([x[:,i]]) for i in range(3)])
        diffusion.append([i*dt,np.average([np.linalg.norm(x[i,:3]-
        COM)**2 for i in range(n)]),np.linalg.norm(COM)])
        if animate_on:
            fig = plt.figure()
            scale = 2*l
            ax = fig.add_subplot(111, projection="3d")
            ax.cla()
            if com_fixed:
                ax.scatter(x[:,0]-np.average(x[:,0]), x[:,1]-
                np.average(x[:,1]),x[:,2]-np.average(x[:,2]),
                cmap = "plasma", alpha=0.3)
            if not com_fixed:
                ax.scatter(x[:,0], x[:,1], x[:,2],
                cmap = "plasma", alpha=0.3)
            ax.set(xlim=(-scale, scale), ylim=(-scale, scale),
            zlim = (-scale, scale))
            ax.set_xlabel("x")
            ax.set_ylabel("y")
            ax.set_zlabel("z")
            fig = plt.figure(dpi=100)
            plt.show()
        x = update(x)
        polarization.append(np.linalg.norm(1/n*sum(x[i,3:]/
        np.linalg.norm(x[i,3:]) for i in range(n))))
diffusion = np.array(diffusion)
if diffusion_plot:
```

```
    plt.plot(diffusion[:,0],diffusion[:,1])
    plt.title("<r^2> of insects about COM ("+model+")")
    plt.show()
    plt.plot(diffusion[:,0],diffusion[:,2])
    plt.title("Norm of COM about origin ("+model+")")
    plt.show()
if polarization_plot:
    plt.plot(diffusion[:,0],polarization)
    plt.title("Polarization of insects ("+model+")")
    plt.show()
if multirun_on:
    diffusion_tot = np.zeros([steps,3])
    polarization_tot = np.zeros(steps)
    diffusion_tot2 = np.zeros([steps,3])
    polarization_tot2 = np.zeros(steps)
    for s in range(runs):
        print(s)
        x = initialize(n,l)
        if model=="MST":
            interactions = insect_mst(x[:,:3])
        elif model=="Delaunay":
            interactions = insect_delaunay(x[:,:3])
        else:
            interactions = [list(filter(lambda x: x!=i,range(n)))
            for i in range(n)]
        diffusion = []
        polarization = []
        for i in range(steps):
            COM = np.array([np.average([x[:,i]]) for i in range(3)])
            diffusion.append([i*dt,np.average([np.linalg.norm(x[i,:3]-
            COM)**2 for i in range(n)]),np.linalg.norm(COM)])
            x = update(x)
            polarization.append(np.linalg.norm(1/n*sum(x[i,3:]/
            np.linalg.norm(x[i,3:]) for i in range(n))))
        diffusion_tot+=np.array(diffusion)
        polarization_tot+=np.array(polarization)
        diffusion_tot2+=np.array(diffusion)**2
        polarization_tot2+=np.array(polarization)**2
    diffusion_tot/=runs
    polarization_tot/=runs
    diffusion_var = diffusion_tot2-runs*diffusion_tot**2
    diffusion_var/=(runs-1)
    diffusion_var = np.sqrt(diffusion_var)
    polarization_var = polarization_tot2-runs*polarization_tot**2
    polarization_var/=(runs-1)
```

```
polarization_var = np.sqrt(polarization_var)
plt.plot(diffusion_tot[:,0],diffusion_tot[:,1],"r-")
plt.plot(diffusion_tot[:,0],diffusion_tot[:,1]-diffusion_var[:,1],"k--")
plt.plot(diffusion_tot[:,0],diffusion_tot[:,1]+diffusion_var[:,1],"k--")
plt.title("<r^2> of insects about COM ("+model+")")
plt.xlabel("Time")
plt.ylabel("<(r-|<r>|)^2>")
plt.show()
plt.plot(diffusion_tot[:,0],diffusion_tot[:,2],"r-")
plt.plot(diffusion_tot[:,0],diffusion_tot[:,2]-diffusion_var[:,2],"k--")
plt.plot(diffusion_tot[:,0],diffusion_tot[:,2]+diffusion_var[:,2],"k--")
plt.title("Norm of COM about origin ("+model+")")
plt.xlabel("Time")
plt.ylabel("|<r>|")
plt.show()
plt.plot(diffusion_tot[:,0],polarization_tot,"r-")
plt.plot(diffusion_tot[:,0],polarization_tot-polarization_var,"k--")
plt.plot(diffusion_tot[:,0],polarization_tot+polarization_var,"k--")
plt.title("Polarization of insects ("+model+")")
plt.xlabel("Time")
plt.ylabel("Polarization")
plt.show()
```

References

[1] Dan Gorbonos (2016) Long-range acoustic interactions in insect swarms: an adaptive gravity model.

[2] Dan Gorbonos, Nir Gov (2017) Stable swarming using adaptive long-range interactions.

[3] Tamas Viscek et al. (2006) Novel type of phase transition in a system of self-driven particles.

[4] Gary Chartrand, Ping Zhang (2005) A First Course in Graph Theory.

[5] Nicolas Giordano (1997) Computational Physics.

[6] Alessandro Attanasi et al. (2014) Collective Behaviour without Collective Order in Wild Swarms of Midges.