5-15-2019

Quantifying Complex Systems via Computational Fly Swarms

Troy Taylor
Linfield College

Follow this and additional works at: https://digitalcommons.linfield.edu/physstud_theses

Part of the Biological and Chemical Physics Commons, Dynamic Systems Commons, and the Non-linear Dynamics Commons

Recommended Citation
Taylor, Troy, "Quantifying Complex Systems via Computational Fly Swarms" (2019). Senior Theses. 43.
https://digitalcommons.linfield.edu/physstud_theses/43

This Thesis (Open Access) is brought to you for free via open access, courtesy of DigitalCommons@Linfield. For more information, please contact digitalcommons@linfield.edu.
Quantifying Complex Systems via Computational Fly Swarms

Troy Taylor

May 15, 2019
THEESIS COPYRIGHT PERMISSIONS

Please read this document carefully before signing. If you have questions about any of these permissions, please contact the DigitalCommons Coordinator.

Title of the Thesis:

Quantifying Complex Systems via Computational Fly Swarms

Author's Name: (Last name, first name)

Taylor, Troy

Advisor's Name

Murray, Joelle

DigitalCommons@Linfield (DC@L) is our web-based, open access-compliant institutional repository for digital content produced by Linfield faculty, students, staff, and their collaborators. It is a permanent archive. By placing your thesis in DC@L, it will be discoverable via Google Scholar and other search engines. Materials that are located in DC@L are freely accessible to the world; however, your copyright protects against unauthorized use of the content. Although you have certain rights and privileges with your copyright, there are also responsibilities. Please review the following statements and identify that you have read them by signing below. Some departments may choose to protect the work of their students because of continuing research. In these cases, the project is still posted in the repository but content will only be accessible by individuals who are part of the Linfield community.

CHOOSE THE STATEMENT BELOW THAT DEFINES HOW YOU WANT TO SHARE YOUR THESIS. THE FIRST STATEMENT PROVIDES THE MOST ACCESS TO YOUR WORK; THE LAST STATEMENT PROVIDES THE LEAST ACCESS. CHOOSE ONLY ONE STATEMENT.

X I agree to make my thesis available to the Linfield College community and to the larger scholarly community upon its deposit in our permanent digital archive, DigitalCommons@Linfield, or its successor technology. My thesis will also be available in print at Nicholson Library and can be shared via interlibrary loan.

OR

_____ I agree to make my thesis available only to the Linfield College community upon its deposit in our permanent digital archive, DigitalCommons@Linfield, or its successor technology. My thesis will also be available in print at Nicholson Library and can be shared via interlibrary loan.

OR

_____ I agree to make my thesis available in print at Nicholson Library, including access for interlibrary loan.

OR

_____ I agree to make my thesis available in print at Nicholson Library only.

Updated April 25, 2018
NOTICE OF ORIGINAL WORK AND USE OF COPYRIGHT-PROTECTED MATERIALS:

If your work includes images that are not original works by you, you must include permissions from the original content provider or the images will not be included in the repository. If your work includes videos, music, data sets, or other accompanying material that is not original work by you, the same copyright stipulations apply. If your work includes interviews, you must include a statement that you have the permission from the interviewees to make their interviews public. For information about obtaining permissions and sample forms, see https://copyright.columbia.edu/basics/permissions-and-licensing.html.

NOTICE OF APPROVAL TO USE HUMAN OR ANIMAL SUBJECTS:

If your research includes human subjects, you must include a letter of approval from the Linfield Institutional Review Board (IRB); see https://www.linfield.edu/faculty/irb.html for more information. If your research includes animal subjects, you must include a letter of approval from the Linfield Animal Care & Use Committee.

NOTICE OF SUBMITTED WORK AS POTENTIALLY CONSTITUTING AN EDUCATIONAL RECORD UNDER FERPA:

Under FERPA (20 U.S.C. § 1232g), this work may constitute an educational record. By signing below, you acknowledge this fact and expressly consent to the use of this work according to the terms of this agreement.

BY SIGNING THIS FORM, I ACKNOWLEDGE THAT ALL WORK CONTAINED IN THIS PAPER IS ORIGINAL WORK BY ME OR INCLUDES APPROPRIATE CITATIONS AND/OR PERMISSIONS WHEN CITING OR INCLUDING EXCERPTS OF WORK(S) BY OTHERS.

IF APPLICABLE, I HAVE INCLUDED AN APPROVAL LETTER FROM THE IRB TO USE HUMAN SUBJECTS OR FROM ANIMAL CARE & USE TO USE ANIMAL SUBJECTS.

Signature: Signature redacted  Date: 5/14/2019

Printed Name: Troy Taylor

Approved by Faculty Advisor: Signature redacted  Date: 5/14/19

Updated April 25, 2018
Abstract

Complexity is prevalent both in natural and in human-made systems, yet is not well understood quantitatively. Qualitatively, complexity describes a phenomena in which a system composed of individual pieces, each having simple interactions with one another, results in interesting bulk properties that would otherwise not exist. One example of a complex biological system is the bird flock, in particular, a starling murmuration. Starlings are known to move in the direction of their neighbors and avoid collisions with fellow starlings, but as a result of these simple movement choices, the flock as a whole tends to exhibit fluid-like movements and form interesting structures. To understand complexity, we chose fly swarms as the system to model. To do this, we utilized stochastic modeling to simulate the movements of individuals, giving them different guiding rules based on both laboratory observation and other models to best produce a realistic model. We hope to compare values of key properties both with other research groups, as well as under varying conditions within our model to find if there is a property that can qualitatively describe if the system is complex or not.
## Contents

3.1 Diffusion .................................................. 13

3.2 Polarization ................................................ 14

3.3 Asymmetry .................................................. 15

4. Results and Analysis ........................................ 17

  4.1 Diffusion .................................................. 17

  4.2 Polarization ................................................ 25

  4.3 Asymmetry vs Swarm Size (Swarm Radius) ............... 26

  4.4 Average Radius vs System Size .......................... 33

5. Conclusions .................................................. 37

6. References .................................................. 39

7. Code .......................................................... 42
1. INTRODUCTION

A system is considered complex if an unexpected property of the system arises from the simple interactions of the individuals that compose the system. This unexpected property is called an *emergent property*. While it is accepted that these features define a complex system, no widely accepted quantitative definition for a complex system exists\(^1\). We hope to probe possible quantitative definitions of complexity via studying computationally simulated fly swarms.

Consider a system of thermodynamic particles close to a phase change. Each particle moving on its own, with its own momenta, while exerting a Van Der Waals force on the others. Raising the temperature, for instance, forces the particles to feel a change in their own momenta. At some temperature, the system as a whole changes drastically, changing the basic properties of what it is. There are many systems that meet the criteria set forth for a complex system, from the stock market to galaxy formations, as well as many biological systems including bird flocks\(^1\), fish schools\(^1\), the growth of plant roots\(^2\), and, of course, insect swarms\(^1,3,4,5,6,7,8\).

Flies that are swarming are doing so for their biological purpose of mating. This means that they cannot drift off from one another and must have some cohesiveness as a whole, but each individual is clearly making a movement choice of its own. What governs these movement choices is not readily apparent, and our models include several different possible options. The flies’ simple interactions with each other
and/or the swarm as a whole contribute to a greater cohesiveness of the swarm. This cohesiveness is the emergent property that improves circumstances for their biological purpose, such as mating. Literature on the complex nature of swarms fails to give a well-defined indicator to determine whether or not the groups of flies are actually swarming. Thus the primary goal for our project is to better define swarming and find a quantitative way to distinguish between swarming and non-swarming states of flies. Various research groups are studying the dynamics of flies in the lab\textsuperscript{1,4}, with which we compare our data to make sure our model realistically reflects the complex nature of the system.

We chose fly swarms because, of the many complex systems, we hoped that it would be easiest to model. However, the hope is that if we can make a model to accurately represent fly swarms, and use that model to understand what is special about them, we could apply the properties of our model to other complex systems. This would allow us to see similarities between diverse complex systems as well as unique aspects. The overall goal of this project is thus to understand complexity as a whole, and fly swarm models are tools to get us there.

1.1 Deterministic vs Stochastic Models

Deterministic systems are systems in which outcomes are dependent on the initial conditions of the system. Most problems in classical mechanics involve these situations; consider, for example, projectile motion. Changing the angle the object is fired has a notable change in the final position, but firing many times with a constant angle always results in the same final position. In contrast, stochastic models are described using a randomness factor as the basis for the movements. In this
case, the same exact conditions can yield vastly different results. While there is a strong understanding of deterministic systems, complex systems require a stochastic approach, as the amount of initial conditions make deterministic models difficult to utilize. Instead, stochastic models account for the biological movement choices allowing for reasonable models of these systems.

A common misconception is to confuse complex systems with chaotic systems. While chaotic systems do exhibit behaviors that can sometimes appear complex, they are still deterministic. Final states from chaotic systems depend on the initial conditions but are much more sensitive to them, in the sense of drastically different outcomes arising from very small changes in initial conditions.

1.2 Complex Systems

Complex systems, despite being prevalent throughout the universe, are not very well understood quantitatively. Complex systems are generally accepted to be a system that is

- Composed of many individuals
- Composed of individuals with simple interactions with one another
- A system in which there is an unexpected emergent property which arises as a result of simple interactions among individuals,

The key point is that there is an emergent property as a result of the individuals interacting. This property is often greater than the sum of its parts since it tends to be some large-scale property while the individual interactions are fairly simplistic. The emergent property is almost always unexpected, in the sense that the individual
interactions do not hint at what the emergent property is. Therefore, given the first two points, one would not be able to deduce the property.

1.2.1 Self-Organized Criticality and Critical States

Complex systems have often been looked at through the scope of what is called self-organized criticality. This idea, used in the study of statistical mechanics and in complex systems that without external influence, states that some systems organize themselves to reach a critical state on their own. Swarms could undergo self organization, since in many ways, fly swarms behave like a system of particles in statistical mechanics. While our model does not probe into the workings of the self organization, we are looking keenly for a critical state in which the system changes between being in a swarm state and a non-swarm state by watching for emergent properties.

1.2.2 Fly Swarms

To study complexity, we are modeling a fly swarm in Matlab stochastically, using different model variations based on observations and theories on fly swarm behavior. Each variation is built upon a base code of $N$ random walkers on lattice points which cannot occupy the same location. A random walker is a point whose movement is completely random.
2. MODELS

2.1 Modeling the System

To simulate the behavior of a system of flies, we started with a random walker model, known as the base model. The flies move completely randomly, with the one caveat that they cannot occupy the same space on the lattice. The base model is used to represent a system of flies that have minimal interactions with one another, which we expect to be a non-swarm state. As illustrated in figure 2.1, we first initialize an array that is $N \times 3$, where $N$ is the number of flies, to get the three-dimensional position of each fly. The code chooses random locations for the flies to start. Each time step, every fly is chosen, in random order, to move, and the flies will select any move at random as long as that position is not already occupied.
2. Models

2.1 Generalized flowchart of the model. In the base model, any potential move is equally likely to be selected while other models weight potential moves based on rules explained in the text.

Regardless of which model we are using, they all follow the same algorithm shown in Figure 2.1.

2.2 Our Models

2.2.1 Global Center of Mass Model

The idea that the flies are drawn to the center of mass of the swarm is proposed by Gorbonos et al., who suggests that midges interact with one another via long-range acoustic stimuli, primarily produced by the flap of each others’ wings. This could imply a global movement choice, in which the fly moves together with the center of mass of the swarm, as opposed to simply following its neighbors. The global center of mass model is simple to weight; we check all possible moves and then measure
their distance from the center of mass. We can then create a new array of these values, normalize them by dividing by the greatest value, multiply it by an array of random numbers ranging from 0 to 1 (which is used as the sole determination for base case movement), and then select the smallest of the array. This results in a movement choice with a weight towards the center of mass while still maintaining the probabilistic nature of the flies, to prevent the system from being deterministic. This weighting applies to the last step in figure 2.1, where we select the move.

### 2.2.2 Local Center of Mass Model

Wang et al. suggested that sudden move choices cascade through the swarm, possibly hinting at the flies being strongly influenced by their nearest neighbors, as opposed to being only affected by the center of mass of the whole collection of flies.\(^8\) The local center of mass model attempts to reflect this behavior. It is similar to the global center of mass model, but instead of using distance from the global center of mass, it scans a certain distance around each fly and creates a center of mass based on the location of flies within that range. Flies with more neighbors, a higher degree,\(^{10}\), are more heavily weighted than a fly on its own. This then gives us a weighting towards the majority of the other nearby flies each time step. Varying the local range of the flies is interesting, and results in different outcomes. As \(R_l \to \infty\), this model becomes identical to the global center of mass model.

### 2.2.3 Global Center of Mass-Velocity Following Model

Flies tend to want to stay with the swarm, so when there is a shift, we expect that the flies on the outside shift as well. By checking how the system moved in the previous time step, we calculate the direction of motion via average velocity
measurements. Each fly has a greater chance of selecting the move most parallel to the system’s previous directional vector. The global center of mass-velocity following model puts an additional, arbitrarily set weight on the move that best aligns with the previous center of mass shift.

2.2.4 Local Velocity Following Model

Similar to the local center of mass model, the local velocity following model considers the idea that the flies are more interested in the motion of their neighbors, as opposed to responding to the average location of the whole swarm. This model takes the idea that the flies move in line with the others, but instead of considering the shift in the swarm’s center of mass, it checks the neighbors. This model simply looks at the position of flies in the previous time step. If there was a fly near the current fly that moved, it will weight the move for the current fly to follow it. To do this, we scan around the fly which is going to move and look at the locations it can move, if a fly was there in the previous time step, the fly will much more likely move along the path which puts it in line with its neighbor’s motion. This model tends to strengthen the swarming behavior of the other models, as it causes flies to follow each other’s motion more often.

2.2.5 Gravity

Laboratory experiments have shown that fly swarms are elongated in the vertical direction, which is attributed to gravity. The influence of gravity can be added to any of the above models by increasing the likelihood of the flies to choose a movement that is along the z-axis. To add gravity to another model, for example, Global Center of Mass and Gravity, each have an array of move choices with certain
probabilities and multiplying them together gives an array of new probabilities for each move choice.

2.2.6 Combining Models

Any combination of models can be achieved following the same approach as is used when adding gravity to a model. As an example of movement selection, consider a situation in which the center of mass is at the origin and a particular fly has three movement choices: 

\[ m_1 = <1, 0, 0>, m_2 = <0, 2, 0>, m_3 = <0, 0, 3> \]

Then the distances from the center of mass are 1, 2, and 3 respectively, so the normalized movement choice vector for global center of mass is \( <\frac{1}{3}, \frac{2}{3}, 1> \). Applying a randomness \( R_n(n = 1, 2, 3) \) to each component, where \( R_n \) ranges from 0 to 1, produces the stochastic nature of the models. The total movement choice array would be \( <\frac{R_1}{3}, \frac{2R_2}{3}, R_3> \). Then adding a third model would be accomplished by multiplying the movement choice vector for that model’s rules to our preexisting vector. To select the move, we pick the minimum of the vector. (a stronger weight is thus a smaller number)

\[ M = \min(< R_1(\frac{1}{3}), R_2(\frac{2}{3}), R_3(1)>) \quad (2.2.1) \]

This example demonstrates that the move closest to the center of mass has a higher probability to be selected, but the factor of \( R_n \) makes the result nondeterministic. In order to add gravity to this model, there is some constant \( (G \leq 1) \) multiplied onto any move along the z-direction, so

\[ M = \min(< \frac{R_1}{3}, \frac{2R_2}{3}, R_3 > \times < 1, 1, G>) \quad (2.2.2) \]

where multiplication of arrays is done component-wise. Thus it is easy to combine the different models, since we only need to multiply their components. The models
that we are using are partitioned into Table 1.
Table 1

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1:</td>
<td>Base Model</td>
</tr>
<tr>
<td>Model 2:</td>
<td>Base Model + Gravity</td>
</tr>
<tr>
<td>Model 3:</td>
<td>Base Model + Local Velocity Following Model</td>
</tr>
<tr>
<td>Model 4:</td>
<td>Base Model + Local Center of Mass Model</td>
</tr>
<tr>
<td>Model 5:</td>
<td>Base Model + Gravity + Local Velocity Following Model</td>
</tr>
<tr>
<td>Model 6:</td>
<td>Base Model + Gravity + Local Center of Mass Model</td>
</tr>
<tr>
<td>Model 7:</td>
<td>Base Model + Local Velocity Following Model + Local COM Model (compound)</td>
</tr>
<tr>
<td>Model 8:</td>
<td>All four rules</td>
</tr>
</tbody>
</table>
3. THEORY

3.0.1 Swarming

As noted before, a complex system is defined by the fact that an emergent property arises from the collective behavior of the individuals. We are defining a fly swarm to be swarming, or in a swarm state if an emergent property is present. This emergent property is a relative cohesiveness as a whole and a fly density that scales appropriately as individuals are added to the system. We consider that an appropriate scaling is that the flies are close enough to undergo their biological purpose of swarming but not becoming so tightly packed as to have collisions. Being able to quantitatively show when a swarm is in a swarm state or not is the overarching goal of the research. Thankfully, Douglas H. Kelley and Nicholas T. Ouellette studied actual flies in the lab\textsuperscript{1} with high speed cameras, tracking the dynamics of the fly system to gather data for the swarm. They found the relation

\[ <r> \propto N^{\frac{1}{3}} \]  

(3.0.1)

Where \(< r >\) is the average radius and \(N\) is the number of flies within the system. What this formula says is that

\[ N = k <r>^3 \]  

(3.0.2)

where \(k\) is some proportionality constant. This can be rewritten to show that

\[ \frac{N}{<r>^3} = k \]  

(3.0.3)
hence the density of flies is constant for all systems that are swarming.

To calculate the average radius of the system, we start with the following formula:

$$< r >_t = \frac{1}{N} \sum_{f=1}^{N} \sqrt{(x_f - x_c)^2 + (y_f - y_c)^2 + (z_f - z_c)^2}$$  \hspace{1cm} (3.0.4)

where subscripts $f$ denote the coordinate of the current fly being checked and the subscript $c$ denotes the coordinate of the current center of mass of the system. The subscript $t$ represents that this is the average radius at timestep $t$. To compare for long term, we will then take the mean of these values over every time step

$$< r > = \frac{1}{T} \sum_{t=1}^{T} < r >_t .$$  \hspace{1cm} (3.0.5)

This value for $< r >$ gives us the overall average radius averaged over both flies and time in one whole run (iteration). Each iteration, we increase the number of flies within the system and then can compare the effect that an increase in individuals has on $< r >$ to see if the power law relationship found by Kelley and Ouellette\textsuperscript{1} exists. We looked at some other properties of the system, such as the diffusion, polarization\textsuperscript{1}, and the asymmetry as a function of swarm size\textsuperscript{5}, as discussed below.

### 3.1 Diffusion

We wanted to look at how much the flies diffuse outward to give us an idea of the dynamics of the system. We monitor the average $(r - r_o)^2$ of the flies to gauge which models are the most/least mobile and this can tell us something about the cohesiveness, provided the center of mass isn’t changing much. Here, $r_o$ is the initial position of the fly, thus a constant value for the particular fly. For models that
are non-cohesive, \((r - r_o)^2\) will continue to rise, since there is nothing keeping them together. We assume a swarm-state will have a plateaued diffusion rate once they become cohesive. This differs from the average radius, which is a measure of how cohesive the swarm is about its own center of mass. Generally speaking, however, the flies moving from their initial position is a good indicator of the diffusion of the system as a whole because the center of mass often does not move much. The total diffusion at time \(t_o\) is calculated as

\[
D(t_o) = \sum_{n=1}^{N} (r_n(t_o) - r_n(0))^2.
\]

(3.1.1)

3.2 Polarization

While the diffusion helps us to see whether or not the flies are moving cohesively or spreading apart, this is not enough to be able to quantify the complete system motion. We also wanted to see how aligned the flies’ velocities are by measuring the polarization(equation 3.2.1). Polarization is the quantifiable value of how in-tune the directions of motions are within a system. For example, in bird flocks where the birds tend to fly in the same direction, there is a polarization of about 1, while less ordered systems have a polarization closer to 0. The Ouellette-Kelley group found that fly swarms in the lab have a polarization of about 0.25\(^1\). Polarization is calculated as

\[
\Phi = \left| \frac{1}{N} \sum_{n=1}^{N} \frac{v_n^*}{v_n} \right|
\]

(3.2.1)

where \(v_n^*\) is the velocity vector of fly \(n\) and \(v_n\) is the magnitude of the vector. So, we can sum up the unit vectors to gauge how ‘in line’ their directions are, since it is normalized by \(\frac{1}{N}\).
3.3 Asymmetry

Asymmetry is the final property we measure in the models. The Gorbonos group\textsuperscript{5} found that in real swarms, the $z$-direction differs from the $x$ and $y$, this is attributed to Earth’s gravitational pull. They found that this influence is seen more strongly in larger swarms so asymmetry grows as a function of swarm radius. We calculated the asymmetry using similar methods to the Gorbonos group, by measuring the ratio of the total moment of inertia about each axis\textsuperscript{5}. Each moment of inertia is denoted as $I_k$ for $k = (x, y, z)$ and defined as:

\[
I_k = \sum_{n=1}^{N} m(k_n - k_o)^2
\]  \hspace{1cm} (3.3.1)

where, $m$ is the ‘mass’ which we define as 1. The moment of inertia tells us the weighted distance about each axis of rotation (cartesian axes). We expect the $\frac{I_x}{I_y}$ value to be nearly 1, since the $x$ and $y$ directions should be equivalent. We compare our data of $\frac{I_z}{I_x}$ with Gorbonos’ findings. Since we look at both individual runs and averages over many runs, we can get the general trend without missing out on possibly interesting individual events.

To make sure that the gravity is making the swarm asymmetric in the $z$-direction as we hope, we know that we want to look at the inertia ratios of $Z$ and $X$, where the inertia is calculated by

\[
I_Z = \sum_{n=1}^{N} m(Z_n - Z_o)^2
\]  \hspace{1cm} (3.3.2)

from equation 3.3.1. $Z_o$ is the $Z$ component of the center of mass, which is calculated by

\[
COM(x, y, z) = \left(\frac{1}{N} \sum_{n=1}^{N} X_n, \frac{1}{N} \sum_{n=1}^{N} Y_n, \frac{1}{N} \sum_{n=1}^{N} Z_n \right)
\]  \hspace{1cm} (3.3.3)
this can in turn be simplified as

$$COM(x, y, z) = \frac{1}{N} \left( \sum_{n=1}^{N} X_n, \sum_{n=1}^{N} Y_n, \sum_{n=1}^{N} Z_n \right) = \frac{1}{N} (X_o, Y_o, Z_o)$$  \hspace{1cm} (3.3.4)$$

so

$$I_Z = N \left( \sum_{n=1}^{N} (Z_n) - \frac{1}{N} \sum_{n=1}^{N} \sum_{n=1}^{N} (Z_o) \right)$$  \hspace{1cm} (3.3.5)$$

since mass is one, so summed over all flies gives us \(N\). Note that \(\sum_{n=1}^{N} (Z_o)\) will result in a constant, the total z-distance from all flies, and summing over a constant \(C\) from 1 to \(N\) gives us \(NC\) so the equation simplifies to

$$I_Z = N \left( \sum_{n=1}^{N} (Z_n) - \sum_{n=1}^{N} (Z_o) \right)$$  \hspace{1cm} (3.3.6)$$

We can do this for all three Cartesian coordinates and then look at the ratios. We expect

$$\frac{I_x}{I_y} \approx 1$$  \hspace{1cm} (3.3.7)$$

as stated earlier, and any inertia ratio pertaining to the z-coordinate should show signs of asymmetry, if the gravity rule is added to the model.
4. RESULTS AND ANALYSIS

Our goal is to quantify what the system of flies is doing for each model. Therefore we first look at three main properties that describe the behaviors fairly well: diffusion of the flies from their original position, to measure how cohesive the system is; the polarization of the system, to measure how “in tune” the flies’ movements are; and the asymmetry in the z-direction (parallel to gravity), to make sure that the gravity adjustment is creating a desired asymmetry.

In addition to the models we are using, the other vital tools in the models include the initial density, which is determined by the value $L$. This sets the maximum $x$, $y$, and $z$ coordinate that each fly can be initialized on. So, having a low $L$ means the flies all start much closer and have a high initial density and the opposite is true for a large $L$. We also have the flies contained within a box, by disallowing any move that would put the fly outside the set wall. The flies cannot move outside the set box to create laboratory-based environmental boundary conditions. We consider these properties for three system sizes: $N$ (number of flies).

4.1 Diffusion

We begin by considering the diffusion of flies in our different models. The base model (Model 1) is used as a reference; we are sure this model is in a non-swarm state. Notice that since the flies do not interact with one another, there cannot be
an emergent property. Figure 4.1 shows typical diffusion results for Model 1.

![Graph showing diffusion results for Model 1.]

**Fig. 4.1**: Diffusion of the base model shows a typical diffusion of random walkers

Since the base model is just random walkers that cannot occupy the same lattice space, this model provides a baseline for comparing other results. It is a good representation for what a non-swarming collection of flies is. We notice that the smaller systems diffuse more because there are more movement choices and anticipate that adding gravity would not have a strong effect on the diffusion rate, which is shown in Figure 4.2.
There is little difference between Figures 4.1 and 4.2 because the addition of gravity only increases the chances of moving up or down equally. This addition to the move select process will not result in a notable change in diffusion rates. In the local velocity following model, model 3, we expect that the flies diffuse even more quickly than in the base model. This behavior can be seen in Figure 4.3; we believe this enhanced diffusion occurs because the flies tend to follow each other outward and the system diffuses more quickly.
Fig. 4.3: Diffusion rate of flies in the local velocity following model.

We expected to see this increased diffusion because we theorized that the local velocity following model will enhance the movement effects of the other models. If the flies are diffusing outward, the local velocity following model increases the likelihood of neighboring flies also diffusing. In the local center of mass model, we anticipated that the flies would diffuse much more slowly, since they are drawn to one another. Figure 4.4 shows this relationship, noting that this effect is stronger in the larger system sizes.
Fig. 4.4: Diffusion rate for the local center of mass model

Since the flies are pulled to one another more strongly when there are more flies, the more populated swarms are more cohesive. Local center of mass systems diffuse more slowly when there are more flies within any given fly’s range. Considering the local velocity following model with the addition of gravity, we expect that it will resemble the data from Figure 4.3, with a slight decrease in diffusion. Figure 4.5 shows exactly this, further enforcing the idea that gravity added to any model will have this weak effect on the diffusion of the model.
Fig. 4.5: Diffusion rate of local velocity following model with gravity.

Similarly to adding gravity to the base model, the addition of gravity to the local velocity following model only slightly decreases the rate that the flies diffuse. This is attributed to the flies selecting $x$ and $y$ moves less frequently. We anticipated that adding gravity to the local center of mass model would have the same effect. Thus we anticipated Figure 4.6 would resemble Figure 4.4 with slower diffusion.
Fig. 4.6: Diffusion rate of local center of mass model with gravity

Again our intuition holds true about the gravity addition to any of our models. Combining the two local models, local center of mass and local velocity following, we anticipated it would be similar to the local center of mass with a stronger cohesiveness. The local velocity averaging model increases the likelihood of flies moving toward one another, initiated by the local center of mass model. We see in Figure 4.7 that this is the case.
Fig. 4.7: Diffusion rate of local center of mass/local velocity following models.

Model 7 resembles model 4 (local center of mass) except with the rate of diffusion decreased because of the local velocity following model. The local velocity following model has the effect of enhancing the effect of the other rules, since the flies tend to follow one another, just as we predicted. Adding gravity to models up to this point has been uninteresting when looking at diffusion, since it always has a weak effect on the model’s diffusion. Figure 4.8 shows the addition of gravity to the local center of mass/local velocity following model.
The addition of gravity to the compound model (model 7) has little effect on the model, likely due to both local velocity and center of mass weightings taking precedence. We also see in Figure 4.8 that model 8 is the least diffusive model. In addition to diffusion, we also wanted to make sure that the flies are polarized similarly to those found naturally.

### 4.2 Polarization

Each model shows a polarization within the range of values shown by the Kelley-Ouellette group with the only exception from the Local Velocity models. In that case, the number of flies is independent of the polarization values. Figure 4.9 is representative of all models, so there was little new information given by these plots. Indeed, we will just note that our models are similar in polarization to that of experiment and thus resemble reality when it comes to the polarization of the swarm.
The polarization sits around 0.2 and has a weak dependence on the number of flies present as seen in Figure 4.9.

---

**Fig. 4.9:** Polarization of Local Center of Mass/Local Velocity

---

### 4.3 Asymmetry vs Swarm Size (Swarm Radius)

We know that for all of the models with gravity, any inertia ratio which includes $I_z$ should deviate from 1. Gorbonos et. al. found that swarms with greater radii (denoted as swarm size) have more asymmetry, and smaller swarms are much more spherical. Figure 4.10 shows the base model’s asymmetry vs swarm radius, which without any directional preference should be, on average, 1 (black line).
4. Results and Analysis

Fig. 4.10: Inertia as a function of swarm radius for the base model.

For the base model, we see that regardless of the size of the swarm, the inertia values including the z-axis are around 1 for all system sizes (N). Adding gravity to the base model, as seen in Figure 4.11, should cause an asymmetry.

Fig. 4.11: Inertia as a function of swarm radius for the base model with gravity.
Adding gravity to the previous model caused the asymmetry to rise, as we would expect. However this effect is seen predominantly in the medium-sized swarms, while large and small swarms maintain asymmetry values close to 1. Local velocity following model, seen in Figure 4.12, does not have a gravity addition and thus should have an average asymmetry of 1.

![Figure 4.12: Inertia as a function of swarm radius for the local velocity following model.](image)

Because there is no gravity in this model, the asymmetry is nearly one. The asymmetry is very similar to that of model 1; note that the local velocity following model allows for larger swarm sizes, so the scales are different but both cases have asymmetry centered around 1. We expect that since the local center of mass model does not have gravity added, it should also follow the trend of the non-gravity models by being centered around 1. Figure 4.13 shows the results from this test.
We see that the systems of fewer flies are stretched more, because the smaller systems diffuse more and thus have greater radii. The particular shapes are uninteresting, though, because on different runs they were different, but the dependency on N was consistent. The largest system size does show an increase in asymmetry with swarm radius. Because of the local center of mass models being more strongly dependent on the system size. The three cases all tend to be around the line of Z=1, though. We anticipate that adding gravity to a model should cause an increase in z-asymmetry, but not have a drastic change in their shape. We see in Figure 4.14 that this is the case; adding gravity to the local velocity following seems to simply raise the average values of z-asymmetry above 1.
Fig. 4.14: Inertia as a function of swarm radius for the local velocity following model with Gravity.

As we would expect, model 5 resembles model 3, just shifted up with more asymmetry due to the addition of gravity. Figure 4.15 shows the local center of mass model with gravity.
Fig. 4.15: Inertia as a function of swarm radius for the Local Center of Mass model with Gravity.

The addition of gravity to the local center of mass model more strongly affected the smaller systems, predominantly $N=20$. Similar to model 4, shown in figure 4.13 the local center of mass causes smaller system sizes to reach greater radii. For the smaller systems the local center of mass rule became less dominant and thus the systems became more asymmetric. Figure 4.16 shows the $z$-asymmetry for the local center of mass/local velocity following model.
4. Results and Analysis

Fig. 4.16: Inertia as a function of swarm radius for the local center of mass/local velocity following model.

The combination of local center of mass and neighbor velocity models resulted in asymmetry that is more sporadic in smaller system sizes and more ordered in larger system sizes. The results of adding gravity to the local center of mass/local velocity following model is shown in Figure 4.17.
Fig. 4.17: Inertia as a function of swarm radius for the Local Center of Mass/Local Velocity model with Gravity.

The asymmetry attributed to gravity is visible as each of the system sizes show values that deviate from 1 much more strongly than in figure 4.16. Strangely, the most asymmetric systems in this model were the smallest. which is the opposite of what is hinted at by Gorbonos et. al.

4.4 Average Radius vs System Size

To probe the relationship between the number of flies and the average radius, we narrowed the models by only considering those with gravity. We looked at base model (Model 2), local center of mass (Model 6), local velocity following model (Model 5), and the combination of local center of mass and local velocity following (Model 8). To test this relation, we looked at the log-log plot so that the slope will give us the power law.
Figure 4.18 shows the results of how the swarm size responds to additional flies in the base model. In the base model we saw that the slope, which represents the power law, is very small. Density in the base model is not kept constant as more flies are added to the system, as Kelley and Ouellette describe in their experiments. We expected this because the base model is our control for a non-swarming model. For the local center of mass model, seen in Figure 4.19, we anticipated that there would be a steeper slope, to show that the swarm better adapts to additional flies.

In the Local Center of Mass model, the slope is comparable to that of the
base model. The results for the local center of mass model showed that there is even a slightly weaker density scaling than even the base model. Figure 4.20 shows the results for the local velocity following model.

![Graph](image)

*Fig. 4.20: Log plot of $< r >$ vs N for the Local Velocity model.*

As with figures 4.18 and 4.19, we found that there is not a density consistency as flies are added to the system. We were not sure what to predict for model 8, as the individual models (local center of mass and local velocity following) didn’t show a drastic difference in slope. Figure 4.21 shows that for model 8, there is a jump in order of magnitude, leading to a much more suitable model for the relation between average radius and system size.
When we combined the local center of mass and local velocity models we found that the slope increased by an order of magnitude. This means that the addition of a fly to the system has a drastically smaller affect on the density of the swarm than in the other models. While we found that there is a slight power law dependence for models 2, 5, and 6, for model 8 (local center of mass, gravity, and local velocity following models), there is an order of magnitude difference in the power, which is very interesting that the models have such synergy. This jump in slope corresponds to the system’s ability to scale to the addition of flies much more efficiently than in the other models and thus the density of the system is changed much less drastically. While there is fluctuation within the graphs, the general slope remains very similar on different runs. We see that the model that appears to deviate the most from the others is model 8, with both neighbor velocity seeking as well as local center of mass seeking.
5. CONCLUSIONS

In the search for a quantitative description for swarming, we found that the model which best matched reality was model 8 (local center of mass, local velocity following, and gravity models) as the scaling properties with system size were most similar to the findings of experimental groups\(^1\). A slow diffusion rate confirms the observation of the simulation that the flies did not simply diffuse outward, but rather stayed more closely with one another, as we would expect in a real fly swarm. The global center of mass model, as well as the center of mass velocity model were quickly ruled out because they appeared, both visually and in the data, more artificial and forced. Regardless of how far away the flies start from each other initially, the global center of mass model always results in them coming together, which is unrealistic. The local versions of both center of mass and velocity seemed to produce more natural motion. This suggests that, at least in the case of our models, the flies’ perception of distance is relatively small. It is still possible that they move based on the air vibrations from the other flies’ wings, but likely only detect the high intensity wave fronts which only result from nearby flies, since the sound wave dissipates as it travels. The unique scaling effect, shown in figure 4.18 of model 8 must arise as a synergistic combination of a fly moving toward its neighbors as well as along with them.

It would be interesting for the relationship between vibration intensity and fly response to be tested more thoroughly and compared to my, as well as other groups’,
findings on the response of flies to their surroundings. While we found which of our models best fits the data we were comparing to, we were unable to definitively give a quantitative definition of swarming. Perhaps the scaling with respect to the number of individuals is the indication of swarming, in which case future research could test this with a model that is assumed to be swarming and compare to a model that is the control group for non-swarm state. If the results are similar to ours, this would show that it is not a single-case and that perhaps the scaling property is, in fact, the best quantitative definition of swarming in the case of flies, and perhaps all complex systems have some form of scaling with respect to the number of individuals present in the system.
6. REFERENCES


Thesis Acceptance

Linfield College

Thesis Title: Quantifying Complex Systems via Computational Fly Swarms

Submitted by: Troy Taylor

Date Submitted: May, 2019

Thesis Advisor: Signature redacted

Dr. Murray

Physics Department: Signature redacted

Dr. Crosser

Physics Department: Signature redacted

Dr. Heath
7. CODE
%Gravity Included in Z–direction

clear all clc

tic

videooo = 0; % 1 for video, 0 for no video
DistbwFlies = 0; %1 for dist calculations, 0 for no (run more quickly)

if videooo == 1
% VIDEO STUFF REMOVE
% video writing stuff
writerObj = VideoWriter('nonaffinity.avi');
writerObj.FrameRate = 35;
open(writerObj);
end

NFLIES = (5:5:150); %Changing system size array
LR = (1:1:length(NFLIES)); %initialize array for local center of mass
for IT = 1:length(NFLIES) %Loop for changing system size
need_to_move = [];
L = 20; %initial density
%nflies = NFLIES(IT); %For comparing systems with different amount of flies
nflies = NFLIES(IT); %number of flies
nsteps = 100; % Number of steps for each iterations
Wall = 200; %boundary
local_range = LR(IT); %Changing Local Range

% If watching simulation, only do one iteration
if videooo == 1
    niterations = 1; % Number of iterations
else
    niterations = 20;
end

save_data = 0; % Yes – 1 or No – 0
% Initialize Variables
time = 0:1:nsteps-1;%create array of time for plot
time = time ’;
position = zeros(nflies,3,nsteps,niterations); % Dimensions:
        X Pos | Y Pos | Z Pos | Time | Iteration
velo = zeros(nflies,nsteps,niterations);%initialize velocity

for iteration = 1:niterations % Generate initial positions
    for each iteration
        [initial_position] = ThreeD_gen(L,nflies); % Generates initial
        positions and get nflies
position(:,:,1,iteration) = initial_position;%initialize position

need_to_move(1,:,1,iteration) = 1:1:nflies;
end

flag = 0;

% VIDEO STUFF REMOVE WHEN NITERATIONS > 1

% plotting initial positions of flies
if videooo == 1
    figure(1)
    plot3(position(:,:,1,1),position(:,:,2,1,1),position(:,:,3,1,1),'*r');
    axis([-20 20 -20 20 -20 20]);
    %axis([-Wall Wall -Wall Wall -Wall Wall]);
    set(gca,'nextplot','replacechildren');
    set(gcf,'Renderer','zbuffer');
    grid on;
    box on;
end

%Differentiate arrays for speed
DiffFrnCoM = zeros(nflies,nsteps,niterations);
GCx = zeros(nsteps,niterations);
GCy = zeros(nsteps,niterations);
7. Code

GCz = zeros(nsteps, niterations);
Cr = zeros(nsteps, niterations);
XChange = zeros(nflies, nsteps, niterations);
YChange = zeros(nflies, nsteps, niterations);
ZChange = zeros(nflies, nsteps, niterations);
Vmag = zeros(nflies, nsteps, niterations);
CofM_Shift = zeros(nsteps, niterations);
O_prime = zeros(niterations, 3);
XcDiff2 = zeros(1, nflies);
YcDiff2 = zeros(1, nflies);
ZcDiff2 = zeros(1, nflies);
DifCoM = zeros(1, nflies);
DC = zeros(1, nflies);
Inertia_Gx = zeros(1, nflies);
Inertia_Gy = zeros(1, nflies);
Inertia_Gz = zeros(1, nflies);
Inertia_SumX = zeros(nsteps, niterations);
Inertia_SumY = zeros(nsteps, niterations);
Inertia_SumZ = zeros(nsteps, niterations);
InertiaSumXY = zeros(nsteps, niterations);
InertiaSumYZ = zeros(nsteps, niterations);
InertiaSumXZ = zeros(nsteps, niterations);
iX = zeros(1, nflies);
iY = zeros(1, nflies);
iZ = zeros(1, nflies);
inertia2 = zeros(nsteps, niterations);
Ave_inertia2 = zeros(1, nsteps);
iSumX = zeros(nsteps, niterations);
iSumY = zeros(nsteps, niterations);
iSumZ = zeros(nsteps, niterations);
iXY = zeros(nsteps, niterations);
iXZ = zeros(nsteps, niterations);
iYX = zeros(nsteps, niterations);
iYZ = zeros(nsteps, niterations);
iZX = zeros(nsteps, niterations);
iZY = zeros(nsteps, niterations);
iXDiff2 = zeros(1, nflies);
iYDiff2 = zeros(1, nflies);
iZDiff2 = zeros(1, nflies);
radius = zeros(nflies, nsteps);
Volume = zeros(nsteps, niterations);
Density = zeros(nsteps, niterations);
CenterVeloMag = zeros(nsteps, niterations);
dVolume = zeros(nsteps, niterations);
PolarizationMagnitude = zeros(nsteps, 1);
XChangeTotal = zeros(1, nsteps);
YChangeTotal = zeros(1, nsteps);
ZChangeTotal = zeros(1, nsteps);
R_s = zeros(nsteps, niterations);
Dist = zeros(nsteps, nflies, niterations);
```matlab
Cx = zeros(nflies, nsteps, niterations);
Cy = zeros(nflies, nsteps, niterations);
Cz = zeros(nflies, nsteps, niterations);
Rx = zeros(nflies, nsteps, niterations);
Ry = zeros(nflies, nsteps, niterations);
Rz = zeros(nflies, nsteps, niterations);

for iteration = 1:niterations
    % Move Flies
    for sec = 2:length(time)%timestep
        need_to_move = 1:1:nflies; % Initialize need to move array
        position(:,:,sec,iteration) = position(:,:,sec−1,iteration);
        % Copy over previous times’ final positions
        while isempty(need_to_move) == 0
            r2 = 0; % Initialize r−squared
            fly = need_to_move(randi(length(need_to_move)))); % Randomly select fly
            clear temp remove
            current_position = position(fly,:,sec,iteration); % Declare current position
            velo(fly,sec,iteration) = randi(2)+1 ;%random velocity of 2−3
            %moves = [velo(fly,sec,iteration),0,0; −velo(fly,sec,
            iteration),0,0;0,velo(fly,sec,iteration),0;0,−velo(fly,sec
```
7. Code

, iteration ), 0; 0, 0, velo ( fly , sec , iteration ) - 1; 0, 0, - velo ( fly , sec , iteration ) + 1; velo ( fly , sec , iteration ) , velo ( fly , sec , iteration ) , 0; - velo ( fly , sec , iteration ) , velo ( fly , sec , iteration ) , 0; velo ( fly , sec , iteration ) , - velo ( fly , sec , iteration ) , 0; - velo ( fly , sec , iteration ) , - velo ( fly , sec , iteration ) , 0; velo ( fly , sec , iteration ) , 0; velo ( fly , sec , iteration ) - 1; velo ( fly , sec , iteration ) , 0, velo ( fly , sec , iteration ) - 1; velo ( fly , sec , iteration ) , 0, - velo ( fly , sec , iteration ) + 1; velo ( fly , sec , iteration ) , 0, - velo ( fly , sec , iteration ) + 1; velo ( fly , sec , iteration ) , 0, velo ( fly , sec , iteration ) - 1; 0, 0, velo ( fly , sec , iteration ) , 0; 0, 0, velo ( fly , sec , iteration ) ; 0, 0, - velo ( fly , sec , iteration ) ] ;

% moves = [ velo ( fly , sec , iteration ) , 0, 0; - velo ( fly , sec , iteration ) , 0, 0; 0, velo ( fly , sec , iteration ) , 0; 0, 0, velo ( fly , sec , iteration ) ; 0, 0, - velo ( fly , sec , iteration ) ] ;

Potential.m anyes = zeros ( Moves , 3 , nflies , nsteps , niterations ) ; %
%Initialize individual coordinate movements

%potential moves in each direction:

\[
\begin{align*}
\text{xp} &= \text{zeros}(1, \text{Moves}); \\
\text{yp} &= \text{zeros}(1, \text{Moves}); \\
\text{zp} &= \text{zeros}(1, \text{Moves}); \\
\text{temp} &= \text{zeros}(\text{Moves}, 3);
\end{align*}
\]

%global center of mass

\[
\begin{align*}
\text{GCx}(\text{sec}, \text{iteration}) &= \text{sum}(\text{position}(::1, \text{sec}, \text{iteration}))/\text{nflies} \\
\text{GCy}(\text{sec}, \text{iteration}) &= \text{sum}(\text{position}(::2, \text{sec}, \text{iteration}))/\text{nflies} \\
\text{GCz}(\text{sec}, \text{iteration}) &= \text{sum}(\text{position}(::3, \text{sec}, \text{iteration}))/\text{nflies}
\end{align*}
\]

%Distance of CofM from origin

\[
\begin{align*}
\text{Cr}(\text{sec}, \text{iteration}) &= \sqrt{\text{GCx}(\text{sec}, \text{iteration})^2 + \text{GCy}(\text{sec}, \text{iteration})^2 + \text{GCz}(\text{sec}, \text{iteration})^2}; \\
\text{CofM}\_\text{Shift}(\text{sec}, \text{iteration}) &= (\text{GCx}(\text{sec}, \text{iteration}) - \text{GCx}(2, \text{iteration}))^2 + (\text{GCy}(\text{sec}, \text{iteration}) - \text{GCy}(2, \text{iteration}))^2 \\
&\quad + (\text{GCz}(\text{sec}, \text{iteration}) - \text{GCz}(2, \text{iteration}))^2;
\end{align*}
\]

\[
\text{O}\_\text{prime}(\text{iteration}, :) = [\text{GCx}(2, \text{iteration}), \text{GCy}(2, \text{iteration}), \text{GCz}(2, \text{iteration})]; \text{origin based on first CofM}
\]

%Relative Center of Mass (COM) Calculation
count = 0;
for e = 1:nflies
%calculate relative COM based off other flies that are within
a
%certain range...what is an appropriate range?
around(fly,sec,iteration) = sqrt((position(fly,1,sec,
    iteration)−position(e,1,sec,iteration)).^2 + (position(fly
    ,2,sec,iteration)−position(e,2,sec,iteration)).^2 + (position(fly
    ,3,sec,iteration)−position(e,3,sec,iteration))^2);
if around(fly,sec,iteration) ⩽ local_range%it should also
count itself
   count = count+1;
Rx(fly,sec,iteration) = Rx(fly,sec,iteration) + position(e,1,
    sec,iteration);%x COM coord
Ry(fly,sec,iteration) = Ry(fly,sec,iteration) + position(e,2,
    sec,iteration);%y COM coord
Rz(fly,sec,iteration) = Rz(fly,sec,iteration) + position(e,3,
    sec,iteration);%z COM coord
else
%essentially don’t do anything
   Rx(fly,sec,iteration) = Rx(fly,sec,iteration);
   Ry(fly,sec,iteration) = Ry(fly,sec,iteration);
   Rz(fly,sec,iteration) = Rz(fly,sec,iteration);
end
%note: if no flies around the fly, COM = it's own position

Cx(fly, sec, iteration) = Rx(fly, sec, iteration)/count;% x center of mass

Cy(fly, sec, iteration) = Ry(fly, sec, iteration)/count;% y center of mass

Cz(fly, sec, iteration) = Rz(fly, sec, iteration)/count;% z center of mass

end

for i = 1:size(moves,1) % Determine each potential move

potential_moves(i,:,fly,sec,iteration) = current_position + moves(i,:);

%Boundary Conditions

if abs(potential_moves(i,1,fly,sec,iteration)) > Wall
    potential_moves(i,:,fly,sec,iteration) = NaN;
else
    potential_moves(i,:,fly,sec,iteration) = potential_moves(i,:,fly,sec,iteration);
end

if abs(potential_moves(i,2,fly,sec,iteration)) > Wall
    potential_moves(i,:,fly,sec,iteration) = NaN;
else
    potential_moves(i,:,fly,sec,iteration) = potential_moves(i,:,fly,sec,iteration);
end
end

if abs(potential_moves(i,3,fly,sec,iteration)) > Wall
    potential_moves(i,:,fly,sec,iteration) = NaN;
else
    potential_moves(i,:,fly,sec,iteration) = potential_moves(i,:,fly,sec,iteration);
end

end

[C, ia, ib] = my_intersect(potential_moves(:,:,:,fly,sec,iteration), position(:,:,:,sec,iteration));

% C is a Nx2 array containing N occupied spots
% ia is a NX1 array containing N row numbers in potential_moves
% that are already occupied (overlap with position rows)
% ib is a 1XNarray containing N row numbers in position
% that might be occupied in the future (overlap with potential_moves rows)

if isempty(ia) == 0 % If there are spots that are occupied
    potential_moves(ia,:,fly,sec,iteration) = NaN; % Mark occupied spots as NaN
end
temp = potential_moves(:,fly,sec,iteration); % Index fly's potential moves
remove = []; 
[remove] = find(isnan(temp(:,1))); % Select rows moves from temp that need to be cleared

nmoves = size(temp,1); % Index number of moves
dis = zeros(1,nmoves);
dis2 = zeros(1,nmoves);
Cap = zeros(1,nmoves);
if nmoves > 1 % If there is more than one move
   for Fly = 1:nflies
      XcDiff2(Fly,sec) = (position(Fly,1,sec,iteration)-GCx(sec,iteration)).^2; % How far x position is from Center of mass x
      YcDiff2(Fly,sec) = (position(Fly,2,sec,iteration)-GCy(sec,iteration)).^2; % How far y position is from Center of mass y
      ZcDiff2(Fly,sec) = (position(Fly,3,sec,iteration)-GCz(sec,iteration)).^2;
      DifFrmCoM(Fly,sec,iteration) = sqrt(XcDiff2(Fly,sec) + YcDiff2(Fly,sec) + ZcDiff2(Fly,sec)); % How far the fly is from center of mass

% * * * * * * * * * *
% Initial values for inertia measurements

inicentdx = (position(Fly,1,sec,iteration)−O_prime(1)).^2;  
How far x position is from initial xCOM

inicentdy = (position(Fly,2,sec,iteration)−O_prime(2)).^2;  
How far y position is from initial yCOM

inicentdz = (position(Fly,3,sec,iteration)−O_prime(3)).^2;  
How far z position is from initial zCOM

DifCoM(Fly) = sqrt(inicentdx + inicentdy + inicentdz);  
magnitude

DC(Fly) = DifCoM(Fly).^2;  %squared magnitude

% Inertia

Inertia_Gx(Fly) = sqrt(ZcDiff2(Fly,sec)+YcDiff2(Fly,sec));
Inertia_Gy(Fly) = sqrt(XcDiff2(Fly,sec)+ZcDiff2(Fly,sec));
Inertia_Gz(Fly) = sqrt(XcDiff2(Fly,sec)+YcDiff2(Fly,sec));

Inertia_SumX(sec,iteration) = sum(Inertia_Gx);
Inertia_SumY(sec,iteration) = sum(Inertia_Gy);
Inertia_SumZ(sec,iteration) = sum(Inertia_Gz);

InertiaSumXY(sec,iteration) = Inertia_SumX(sec,iteration)./
Inertia_SumY(sec,iteration);
InertiaSumYZ(sec,iteration) = Inertia_SumY(sec,iteration)./
Inertia_SumZ(sec,iteration);
InertiaSumXZ(sec, iteration) = Inertia_SumX(sec, iteration) / Inertia_SumZ(sec, iteration);

iX(Fly) = sqrt(inicentdy + inicentdz); % inertia w/ x axis
iY(Fly) = sqrt(inicentdx + inicentdz); % inertia w/ y axis
iZ(Fly) = sqrt(inicentdx + inicentdy); % inertia w/ z axis

% from initial COM

inertia2(sec, iteration) = sum(DC);
Ave_inertia2(sec) = sum(inertia2(sec,:))/niterations;

iSumX(sec, iteration) = sum(iX);
iSumY(sec, iteration) = sum(iY);
iSumZ(sec, iteration) = sum(iZ);

% Inertia Ratios

iXY(sec, iteration) = iSumX(sec, iteration)/iSumY(sec, iteration);
iXZ(sec, iteration) = iSumX(sec, iteration)/iSumZ(sec, iteration);
iYX(sec, iteration) = iSumY(sec, iteration)/iSumX(sec, iteration);
iYZ(sec, iteration) = iSumY(sec, iteration)/iSumZ(sec, iteration);
iZX(sec, iteration) = iSumZ(sec, iteration)/iSumX(sec, iteration);

iZY(sec, iteration) = iSumZ(sec, iteration)/iSumY(sec, iteration);

%For inertia, need distance-squared from initial CoM

iXDiff2(Fly) = (position(Fly,1,sec,iteration)−GCx(2,iteration)).^2;

iYDiff2(Fly) = (position(Fly,2,sec,iteration)−GCy(2,iteration)).^2;

iZDiff2(Fly) = (position(Fly,3,sec,iteration)−GCz(2,iteration)).^2;

end

% * * * * * * * * * * * * * * * * * * * * * * * * * *

Art = [];% temporary array to ignore stragglers

% ignoring flies outside of scope of swarm

Art = sort(DifFrmCoM(:,sec,iteration));

for art = 1:length(Art)

if Art(art) > 2*mean(DifFrmCoM(:,sec,iteration))

Art(art) = NaN;

end

end
rid = [];

[din] = find(isnan(Art));

Art(din) = [] ; %get rid of stragglers

for a = 1:length(Art)
    Art2(a,sec) = Art(a);
end

Art3(sec) = nnz(Art2(:,sec)); %how many nonzero [how many

    flies in actual swarm]

flux(sec) = abs(Art3(sec)−Art3(sec−1)); %how many flies left

    or joined the actual swarm]

R_s(sec,iteration) = (sum(Art))/(nflies); %average radius

%CenterMovement(sec,:) = [GCx(sec)−GCx(sec−1),GCy(sec)−GCy(

    sec−1),GCz(sec)−GCz(sec−1)];

G = ones(Moves,1);

for j=1:nmoves
    Newton = ones(Moves,1); %re−initialize arrays

xp(j) = temp(j,1); %x coords of potential moves

yp(j) = temp(j,2); %y coords of potential moves
zp(j) = temp(j,3); % z coords of potential moves

% Studies show flies tend to move in the z-direction less

% Weight on Center of Mass Velocity;

VelWt = 0.8; % weighting put on movements towards center of mass movement

CenterVelo = [XChangeTotal(sec-1), YChangeTotal(sec-1), ZChangeTotal(sec-1)];

CenterVeloPos = [abs(XChangeTotal(sec-1)), abs(YChangeTotal(sec-1)), abs(ZChangeTotal(sec-1))];

[Value, Index] = max(abs(CenterVelo));

if CenterVelo(Index) > 0 % If the center of motion's greatest direction is positive

% weight move towards greatest motion in the same direction

if Index == 1

Newton(1) = Newton(1)*VelWt; % First choice is +x
end

if Index == 2

Newton(3) = Newton(3)*VelWt; % Third choice is +y
end

if Index == 3

Newton(5) = Newton(5)*VelWt; % Fifth choice is +z
end
if CenterVelo(Index) < 0
if Index == 1
Newton(2) = Newton(2) * VelWt; % Second choice is -x
end
if Index == 2
Newton(4) = Newton(4) * VelWt; % Fourth choice is -y
end
if Index == 3
Newton(6) = Newton(6) * VelWt; % Sixth choice is -z
end
end
Newton(remove) = NaN;
Cap(remove) = NaN;
dist(j) = sqrt((xp(j) - GCx(sec))^2 + (yp(j) - GCy(sec))^2 + (zp(j) - GCz(sec))^2); % Distance of each potential move from COM

dist2a(j) = dist(j) * ((xp(j) - GCx(sec))^2 + (yp(j) - GCy(sec))^2);
% Distance times dist from z-axis for weighting for grav
% dist2a(j) = (zp(j) - GCz(sec)) * sqrt((yp(j) - GCy(sec))^2 + (zp(j) - GCz(sec))^2);
dist2(j) = dist(j)^2;
disr(j) = sqrt((xp(j) - Cx(fly, sec))^2 + (yp(j) - Cy(fly, sec))^2 + (zp(j) - Cz(fly, sec))^2); % Relative center of mass
weighting
disr2(j) = ((xp(j)−Cx(fly,sec)).^2 + (yp(j)−Cy(fly,sec)).^2 +
(zp(j)−Cz(fly,sec)).^2); %relative center of mass
weighting

if DistbwFlies == 1

%Distance between flies distribution
F = 1;
ig = 1;
for ic = F:nflies−1
for id = (F+1):nflies
Dist(sec,ig,iteration) = sqrt((position(ic,1,sec,iteration) −
position(id,1,sec,iteration))^2 + (position(ic,2,sec,iteration) −
position(id,2,sec,iteration))^2 + (position(ic,3,sec,iteration) −
position(id,3,sec,iteration))^2);
ig = ig + 1;
end
F = F+1;
end
Count_value = (nflies)*(nflies−1)/2; %number of distances
counted (all above or below diagonal)
dist_tot(sec,iteration) = sum(Dist(sec,: ,iteration));
dist_av(sec,iteration) = dist_tot(sec,iteration)/Count_value;
end
% * * * * * * * * *
% * * * * * * *

% Attempting to weight toward neighboring motions

D = 1;

Nweight = 0.25; % weighting toward neighbor directions

if sec > 1

Neigh = ones(nflies, length(moves));

for midge1 = D:nflies−1

for midge2 = (D+1):nflies

if le(abs(position(midge1,1,sec,iteration)−position(midge2,1,
sec−1,iteration)),2) % if midge2 was neighboring in the x−direction

Checkx(:) = position(midge2,1,sec,iteration) − position(
midge2,1,sec−1,iteration); % check which way midge2 went

if Checkx > 0 % if neighbor moved up in the x

Neigh(midge1,1) = Nweight∗Neigh(midge1,1); % weight move 1 [i.e. positive x move]

end

if Checkx < 0 % if neighbor moved down in the x

Neigh(midge1,2) = Nweight∗Neigh(midge1,2); % weight move 2 [i.e. negative x move]

end

end

end

if le(abs(position(midge1,2,sec,iteration)−position(midge2,2,
sec−1,iteration)) %if midge2 was neighboring in the y−direction

Checky = position(midge2 ,2,sec,iteration) − position(midge2 
 ,2,sec−1,iteration); %check which way midge2 went
if Checky > 0 %neighbor moved up in the y
Neigh(midgel,3) = Nweight∗Neigh(midgel,3);
end
if Checky < 0 %if neighbor moved down in the y
Neigh(midgel,4) = Nweight∗Neigh(midgel,4); %weight move 4 [i.e. negative y move]
end

if le(abs(position(midgel,3,sec,iteration)−position(midge2,3,sec−1,iteration)),2) %if midge2 was neighboring in the z−direction

Checkz = position(midge2,3,sec,iteration) − position(midge2 
 ,3,sec−1,iteration); %check which way midge2 went
if Checkz > 0 %neighbor moved up in the z
Neigh(midgel,5) = Nweight∗Neigh(midgel,5);
end
if Checkz < 0 %if neighbor moved down in the z
Neigh(midgel,6) = Nweight∗Neigh(midgel,6); %weight move 6 [i.e. negative z move]
end
end
end
D = D+1;
end

% inertia and continued motion of flies
move = position(fly,: ,sec,iteration) - position(fly,: ,sec-1,iteration);
ve(fly,sec,iteration) = norm(move);
% ve is essentially the velocity of the last move
Foreward.Weight = 0.5; % more likely to continue direction
Backwards.Weight = 2; % much less likely to turn back immediately

% x
if position(fly,1,sec,iteration) - position(fly,1,sec-1,iteration) > 0 % if fly moved +x last turn
Newton(1) = Newton(1)*Foreward.Weight;
Newton(2) = Newton(2)*Backwards.Weight;
end

if position(fly,1,sec,iteration) - position(fly,1,sec-1,iteration) > 0 % if fly moved +x last turn
Newton(2) = Newton(2)*Foreward.Weight;
Newton(1) = Newton(1)*Backwards.Weight;
end

%y
if position(fly,2,sec,iteration) − position(fly,2,sec−1,
     iteration) > 0 %if fly moved +x last turn
Newton(3) = Newton(3)∗Forward.Weight;
Newton(4) = Newton(4)∗Backwards.Weight;
end

if position(fly,2,sec,iteration) − position(fly,2,sec−1,
     iteration) > 0 %if fly moved +x last turn
Newton(4) = Newton(4)∗Forward.Weight;
Newton(3) = Newton(3)∗Backwards.Weight;
end

%z
if position(fly,3,sec,iteration) − position(fly,3,sec−1,
     iteration) > 0 %if fly moved +x last turn
Newton(5) = Newton(5)∗Forward.Weight;
Newton(6) = Newton(6)∗Backwards.Weight;
end

if position(fly,3,sec,iteration) − position(fly,3,sec−1,
     iteration) > 0 %if fly moved +x last turn
Newton(6) = Newton(6)∗Forward.Weight;
Newton(5) = Newton(5)∗Backwards.Weight;
end
Grav = [1,1,1,1,0.5,0.5]; %gravity increases chance of z-directional choices

ndr = (1/max(disr2))*(disr2(:,.')); %normalize "disr"

G_min = 0;%Lowest possible G value

G_max = 1;%highest possible G value

G(j) = (G_max-G_min).*rand()+G_min; %Swarm randomness factor

G(remove) = NaN;

Cap(j) = G(j).*Neigh(fly,j).*dis2(j); %determine which weights to apply for move

end

[v,w]=min(Cap); %select weighted random move

if isnan(temp(:)) %If fly can’t move, don’t move

position(fly,:,sec,iteration) = position(fly,:,sec,iteration)

;

else

move_select=w;%select that move

position(fly,:,sec,iteration) = temp(move_select,:); % Move fly

end

radius(fly,sec) = sqrt((position(fly,1,sec,iteration)-GCx(sec)
7. Code

\[ r_{av}(sec,\text{iteration}) = \frac{\text{sum(radius(:,sec))}}{nflies}; \]

\%Polarization Calculations

\[ V = \text{zeros}(3, nflies); \]

\text{for FLI = 1:nflies}

\[ \text{XChange(FLI,sec,iteration)} = \text{position(FLI,1,sec,iteration)} - \text{position(FLI,1,sec-1,iteration)}; \]

\[ \text{YChange(FLI,sec,iteration)} = \text{position(FLI,2,sec,iteration)} - \text{position(FLI,2,sec-1,iteration)}; \]

\[ \text{ZChange(FLI,sec,iteration)} = \text{position(FLI,3,sec,iteration)} - \text{position(FLI,3,sec-1,iteration)}; \]

\[ V(:,FLI) = [\text{XChange(FLI,sec,iteration)}, \text{YChange(FLI,sec,iteration)}, \text{ZChange(FLI,sec,iteration)}]; \]

\[ \text{Vmag(sec,FLI,iteration)} = \sqrt{\text{XChange(FLI,sec,iteration)}.^2 + \text{YChange(FLI,sec,iteration)}.^2 + \text{ZChange(FLI,sec,iteration)}.^2}; \]

\[ \text{unitV(:,FLI)} = \text{V(:,FLI)}/\text{Vmag(sec,FLI,iteration)}; \]

\text{end}

\[ \text{Polarization} = \text{abs}(1/nflies} \times [\text{sum(unitV(1,:))}, \text{sum(unitV(2,:))}, \text{sum(unitV(3,:))}]; \]

\[ \text{PolarizationMagnitude(sec,iteration)} = \sqrt{\text{Polarization(1)}^2 + \text{Polarization(2)}^2 + \text{Polarization(3)}^2}; \]
XChangeTotal(sec) = sum(XChange(:,sec,iteration));
YChangeTotal(sec) = sum(YChange(:,sec,iteration));
ZChangeTotal(sec) = sum(ZChange(:,sec,iteration));

need_to_move(need_to_move==fly) = []; % Remove fly from need to move list
dis = []; % reset distance for determining next fly's position
av_dist = [];
weighting = []; % reset weighting for next fly

elseif nmoves == 1 % If there is one move
position(fly,:,sec,iteration) = temp; % Move fly
need_to_move(find(need_to_move==fly)) = []; % Remove fly from need to move list

else
position(fly,:,sec,iteration) = position(fly,:,sec-1,iteration); % Keep fly at position at previous time step
need_to_move(need_to_move==fly) = []; % Remove fly from need to move list
end

% Calculate stuff for diffusion if fly moves
xdiff2 = (position(fly,1,sec,iteration)-position(fly,1,1,iteration))^2;
ydiff2 = (position(fly, 2, sec, iteration) - position(fly, 2, 1, iteration)).^2;
zdifff2 = (position(fly, 3, sec, iteration) - position(fly, 3, 1, iteration)).^2;
r2 = r2 + xdiff2 + ydiff2 + zdiff2;
end

rx(sec, iteration) = sum(position(:, 1, sec, iteration));
ry(sec, iteration) = sum(position(:, 2, sec, iteration));
rz(sec, iteration) = sum(position(:, 3, sec, iteration));
r2av(sec, iteration) = r2 / nflies;
if videooo == 1
% VIDEO WRITING STUFF REMOVE IF NITERATION > 1
swarm = plot3(position(:, 1, sec, iteration), position(:, 2, sec, iteration), position(:, 3, sec, iteration), '*k');
title(sprintf(['Time: ', num2str(sec)]));
frame = getframe;
writeVideo(writerObj, frame);
end
calc(sec, iteration) = max(Art);
Volume(sec, iteration) = ((calc(sec, iteration)).^3); % proportional to r^3
dVol(sec, iteration) = Volume(sec, iteration) - Volume(sec-1, iteration);
end

progress = 100*iteration/niterations;

av_V = mean(dVol,1);

iteration

progress;

av_V;

end

rav2 = mean(rav,2);

Time = [1:1:length(NFLIES)];

av_r2av(:,IT) = mean(r2av,2); % Averages all r-squared average values

figure(60)

plot(time,rav2,'-')

hold on

RR(IT) = mean(rav2,1);

end

figure(70)

hold on

plot(NFLIES,RR,'-')

avRad = mean(av_r2av,1);

AveragePolo = mean(PolarizationMagnitude,2);

figure(71)

hold on

plot(log(NFLIES),log(RR),'-');
xlabel('Log(Number of Flies)')
ylabel('Log(Average Radius)')
title('Local Velocity/Local Center of Mass')
flux(2) = 0;
ave_V = mean(dVol,2);
AveragePolo = mean(PolarizationMagnitude,2);
R_av = mean(Rs,2);
R = mean(R_av,1);
figure(100)
plot(LR,avRad,'-')
meaniXY = mean(iXY,2);
meaniXZ = mean(iXZ,2);
meaniYX = mean(iYX,2);
meaniYZ = mean(iYZ,2);
meaniZX = mean(iZX,2);
meaniZY = mean(iZY,2);
MEAN = mean(InertiaSumXZ,2);
if DistbwFlies == 1
avdis = mean(dist_av,2);
end
runtime = toc;
\% * * * * *

STD_P = std(AveragePolo);

STD_D = std(av_r2av);

STD_I = std(iXZ);

STD_Ixz = mean(STD_I);

if save_data == 1
\%
 Save Data

directory = [num2str(nflies), ' Flies '];

mkdir(directory);

save([num2str(nflies), '_flies_position_v7.3_v2.mat'], 'position', '-v7.3'); \% Might be needed for nflies > 800
save([num2str(nflies), '_flies_position_v2.mat'], 'position');

save([num2str(nflies), '_flies_av_r2av_v2.mat'], 'av_r2av');

save([num2str(nflies), '_flies_av_slope_v2.mat'], 'av_slope');

save([num2str(nflies), '_flies_av_intercept_v2.mat'], 'av_intercept');

save([num2str(nflies), '_flies_av_rmse_v2.mat'], 'av_rmse');

save([num2str(nflies), '_flies_av_best_fit_v2.mat'], 'av_best_fit');

save([num2str(nflies), '_flies_data_v2.mat'], 'data');

end

if videooo == 1

close(writerObj);

end