A Dimensional Reduction Analysis of the Self-Propelled Particle Model

Princeton University

Nick Lavrov

Advisor: Professor Ioannis G. Kevrekidis

April 27, 2015

Submitted in partial fulfillment of the requirements for the degree of
Bachelor in Science in Engineering

Department of Chemical and Biological Engineering

This paper represents my own work in accordance with University regulations.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

_____

Nick Lavrov

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Nick Lavrov

Princeton University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Dedication

This work, along with the entirety of my Princeton education, is dedicated to my mother who sacrificed so much to raise me, to my stepfather who worked so hard to teach me, and to my sister, my favorite sister, who is always with me.

Acknowledgements

Completing my senior thesis was a group effort. I would like to thank my advisor Professor Ioannis Kevrekidis for steering my work in the right direction whenever I turned down a dead end. I owe thanks to Felix Kemeth, a graduate student who found the time to introduce me to diffusion maps, and Assimakis Kattis, a classmate of mine, for showing me a few sources and generally being on my side.

I would never have dreamed of coming to Princeton University were it not for the efforts of my mother, Rose Spano, and my stepfather, Brian Hegarty. I truly owe my education to their tireless efforts to make me who I am today. I hope those efforts were not in vain. My wonderful sister, Natalie Lavrov, has also showed me support when I needed it and funny Internet videos when I definitely needed it. Thank you so much.

I owe so much to my best friends, my roommates. Yegor Chekmarev, I can truly say it has been an honor working along side of you. Arjun Dube, you have never failed to make me laugh, and thank you for driving me to Wall that one time. Chester Dubov, thank you for making me look sane by comparison. Utsarga Sikder, thank you for being the voice of reason. Junya Takahashi, thank you for providing me with strategic study breaks. Playing Tower Fall last night was fun. Gavin Cook, even though you do not live with us, you were an important presence during my hours of work. Our door is always open.

Alison Itzkowitz, I am so happy I met you this year. Thank you for being a great Fourth Course cook and for supporting me. Christina Chica, we go back to Princeton In Beijing, but I think this year we have truly grown closer. Thank you for being someone to talk to and for bringing me dinner when I could not leave my bed. Annie Chen, you are an amazing person in every sense of the word, and thank you for bringing me delicious snacks. Greg Jo, we have had a lot of fun together. I wish I could list all my other friends in Princeton. Thank you everyone for your company and your support.

Harsh Patel, I remember when we were two naïve little guys on the first day of high school. Well, look at us now. Thank you for sticking with me for so long. Emilia Iwaszkowska, spending time with you is always something I enjoy. You provided me with a great connection outside of Princeton.

Olivia Watson, you made me so happy, even during my crazy work schedule. Meeting you is the best thing that has ever happened to me. I cannot imagine myself without you. Thank you so much.

Finally, I must give thanks to the entire staff of Princeton University. The Financial Aid Office ensured I could attend here. My professors gave me the best education. The custodial staff kept the hallways clean. My Italian teachers and my Chinese teachers taught me new ways to express myself. Grazie e 谢谢. The dining hall staff made my favorite food. I cannot forget to mention the institutions within the University that never gave me homework: Lobster Club, Colonial Club, Terrace F Club, and WPRB. I leave Princeton University as a changed man.

Stellis Aequus Durando,

Nick Lavrov

Abstract

Locusts continue to have devastating effects on the modern world. The swarming behavior of locusts emerges from the individual motion of each locust. How complex organized motion arises within animals represents a large area of research. The self-propelled particle (SPP) model developed by Vicsek *et al*. (1995) successfully creates swarming behavior similar to that observed by locusts studied by Buhl *et al*. (2006). The SPP model creates a high dimensional data set of the positions and velocities of each locust at each time step. A better understanding of the dynamics of the SPP model could be used to control swarms of locusts before they destroy crops. Dimensional reduction techniques are used to discover if there exists an underlying substructure within the data. Both principal component analysis (PCA) and diffusion map analysis (DMA) identify the average velocity of the locusts, the alignment, as an important variable when tracking the switching behavior of particles on a one-dimensional domain. Further analysis identifies the relationship between the probability of switching and the alignment. The alignment of the swarm can be forced to switch by artificially changing the velocity of individual locusts to move against the swarm. The minimum number of locusts required to force a switch, known as the critical number, is found as a function of the alignment. When the alignment is near 0, setting 6 locusts out of 30 (20% of the population) is enough to result in a change in alignment 90% of the time. Future work could expand on these findings to see if they apply *in vivo*.

Contents

List of Figures

1 Introduction

On a hot August day in 2014, a swarm of millions of locusts descended on Madagascar's capital city of Antananarivo, slowing down traffic and blotting out the skies. This swarm was part of a plague that began in April 2012, when a heat wave disturbed the insects and drove them from their typical rural habitat. The locusts have threatened the food security of over 13 million people, or almost 60% of the population, and Madagascar's emergency plan, projected to continue until 2016, will cost more than $41 million [1].

Locust swarming is just one of the many types of collective motion observed in nature. Patterns of behavior within ant colonies resemble that of a neural network, with coordinated waves of activity propagating through the colony [2]. Small fish travel in schools that rapidly react to attacks made by predators [3]. Small birds form flocks through which information can be transferred in a scale-free manner [4]. The macroscopic emergent behaviors of these systems are the result of microscopic individual behaviors of their components. Models have been developed to recreate these behaviors in computer simulations from a set of simple rules [5]. One such model is the self-propelled particle (SPP) model pioneered by Vicsek *et al*. (1995) in which the particles follow just one rule: at each time step a given particle "assumes the average direction of motion of the particles in its neighborhood of radius r with some random perturbation added" [6]. Simulations of the SPP model in one and two dimensions reveal cohesive swarms that periodically split or change direction in the absence of external perturbations [6-9].

Modeling the swarming behavior of desert locusts, *Schistocerca gregaria*, could point to ways to control their devastating social and economic impact. Bands of wingless locust nymphs can cover many kilometers. These bands form when docile locusts in the docile "solitarious" phase switch to the aggressive "gregarious" phase. These bands persist until the locusts become adults and take flight, creating the swarms of locusts as seen in Antananarivo [10-13].

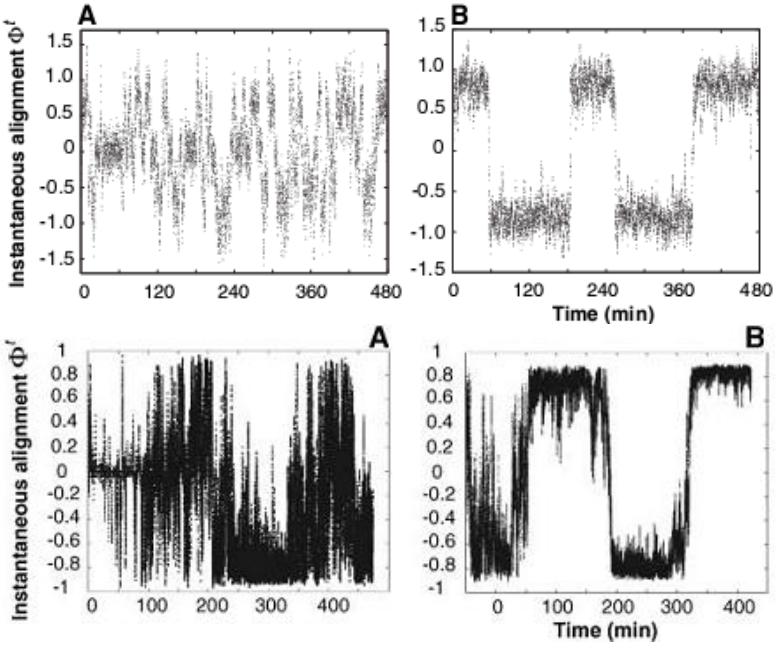


Figure 1.1: The alignment of the SPP model (top) and experimental results (bottom) for trials with 3 locusts (A) and 11 locusts (B). The model and experimental results have similar behaviors both in value and in switching times. Figures are from Buhl *et al*. (2006) [14].

In order to develop an accurate computational model for these locusts in particular, experimental data must be supplied for comparison. Buhl *et al*. (2006) tracked the motion of third-instar desert locust nymphs along a ring-shaped container over a long period of time. From this data, the orientation of a locust was calculated at each time step by measuring the smallest angle between one line connecting the locust's two consecutive positions and one line connecting the center of the circle to the locust's first position. The alignment was defined as the average of all orientations for each time step

[14]. For the one-dimensional SPP model this measure of alignment is analogous to the average velocity of all locusts at each time step. **Figure 1.1** shows the correspondence of the two quantities. The experimental data and the SPP model produce similar behavior.

The behavior of the alignment of the locust swarm also matches closely with the behavior of stochastic differential equations (SDEs) [15]. By using equation free techniques as in [15-17], the SDE behind the SPP model can be approximated (see **Fig. 1.2**). In such a way, a quantity similar to the alignment can be simulated by one less computationally expensive SDE as opposed to the entire SPP model.

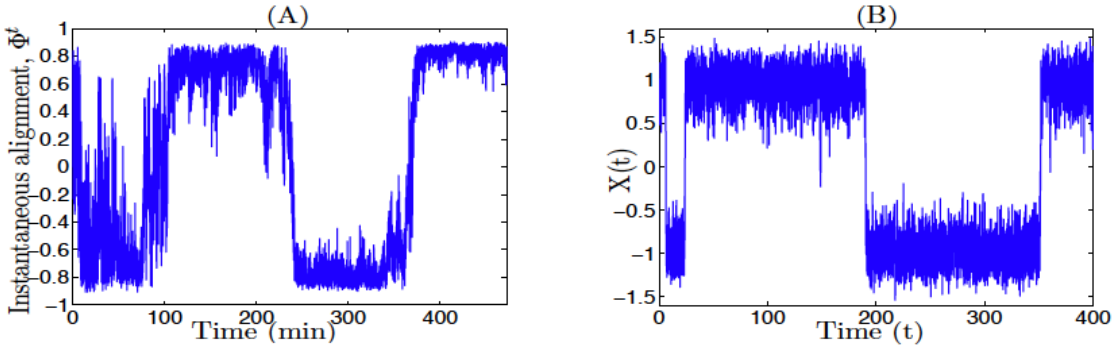


Figure 1.2: Experimental results from Buhl *et al.* (2006) [14] (A) compared to the numerical solution of the SDE found by Yates (2007) [15] (B). Similar behavior is observed in the tendency to settle at a given value and the frequency of switches.

Another way to identify interesting information about group dynamics is to apply dimensional reduction techniques to the data generated by the SPP model. Dimension reduction techniques extract meaningful lower dimensional structures and correlations from higher dimensional datasets [18]. In **Fig. 1.3**, the underlying two-dimensional manifold of the three-dimensional data is uncovered through a dimensional reduction technique. Face recognition is a common application of dimensional reduction techniques [19]. The two techniques used in this paper are principal component analysis (PCA) and diffusion map analysis (DMA).

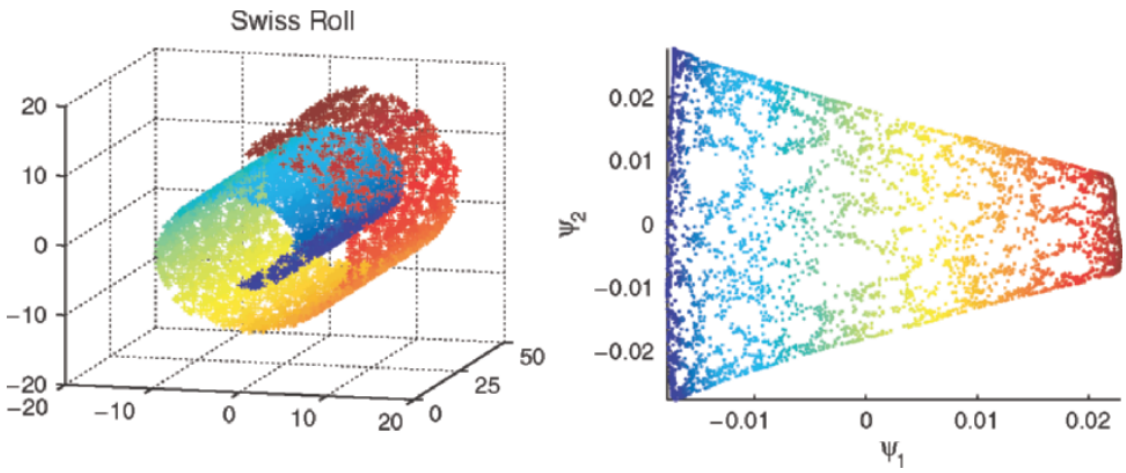


Figure 1.3: The underlying two-dimensional manifold (right) of the three-dimensional Swiss roll data is revealed through diffusion map analysis. Figure from Nadler *et al*. (2008) [26].

PCA is a linear dimensional reduction technique that uses singular value decomposition of a matrix of data observations to identify the directions of greatest variance within the data. By taking the first *p*-directions, a *p*-dimensional representation of the data set is produced that accounts for the greatest possible amount of variance of the data [20]. PCA has applications in finding low-dimensional descriptions of turbulent flow [21], complex vibrations [22], and damage detection [23]. PCA is limited, however, by its inability to identify nonlinear structures within data. The steps of PCA are given in more detail in the Section 3.1.

DMA is a more generalized form of PCA that can be applied to nonlinear datasets [24]. Instead of relying on the Euclidean distance between points as in PCA, DMA uses the diffusion distance between points. The diffusion distance can be imagined as the similarity of two points in space with their connectivity, and the connectivity is related to the probability of taking a random walk along the data from one point to the next [25]. The exact algorithm is given in the Section 3.2.

2 Background

2.1 The Self-Propelled Particle Model

While the exact underlying mechanics of locust swarming are not fully known, the self-propelled particle (SPP) model developed by Vicsek *et al*., Czirok *et al*., and Yates [6, 7, 15] provides a good approximation. The emergent behavior of the SPP model matches what is seen in the lab. Even though the rules all work on an individual level, collective behavior arises in a prominent manner. The rules are applied at each time step and are simple in their implementation. The presentation of following steps for the SPP model follows that given by Szeto (2009) [27].

In the SPP model, a series of points with given positions and velocities act as the locusts. There is a set interaction radius that defines how far each point can "see" other points and interact. The model given here is one-dimensional in a limited circular length domain. At each time step, the position and velocity of the points are updated as follows:

$$x_i(t + \Delta t) = x_i(t) + u_i(t)(\Delta t) \tag{2.1}$$

$$u_i(t + \Delta t) = u_i(t) + [G(\langle u(t)\rangle_i) - u_i(t)]\Delta t + Q \tag{2.2}$$

$$\langle u(t)\rangle_i = \frac{1}{j}\sum_{j \in IR} u_j(t) \tag{2.3}$$

Where x_i and u_i represent the position and velocity, respectively, of locust i. The discretized time step is Δt. The sum over $j \in IR$ represents all locusts within the interaction radius of locust i.

Equation 1 updates the position of each particle based on its velocity. Equation 2.2 updates the velocity of each particle by adding a function of the average velocity of all particles (given in Equation 3) within the interaction radius and some noise term.

The function G given in Equation 2.2 is expanded in Equation 2.4. It represents how much the velocity of the individual locust is weighted compared to the average velocity of the surrounding locusts.

$$G(z) = \frac{1}{1+\beta}\begin{cases} z+\beta, & z > 0 \\ 0, & z = 0 \\ z-\beta, & z < 0 \end{cases} \tag{2.4}$$

Where β is a parameter set by the user. In all simulations, β is set to 1 for simplicity. The noise term $Q(x)$ follows a simple uniform distribution given below:

$$Q(x) = \begin{cases} 0, & x < -\mu/2 \\ 1/\mu, & -\mu/2 < x < \mu/2 \\ 0, & x > \mu/2 \end{cases} \tag{2.5}$$

The mean is 0 and the variance is $\mu^2/12$. As used in Yates [15], $\mu(\Delta t)$ is set to $3(\Delta t)^{.5}$ in order to replicate the behavior shown the experimental model developed by Buhl *et al* [14]. The SPP model uses a time step of 1 in all implementations in this paper, so the noise term has a mean of 0 and a variance of 1/4.

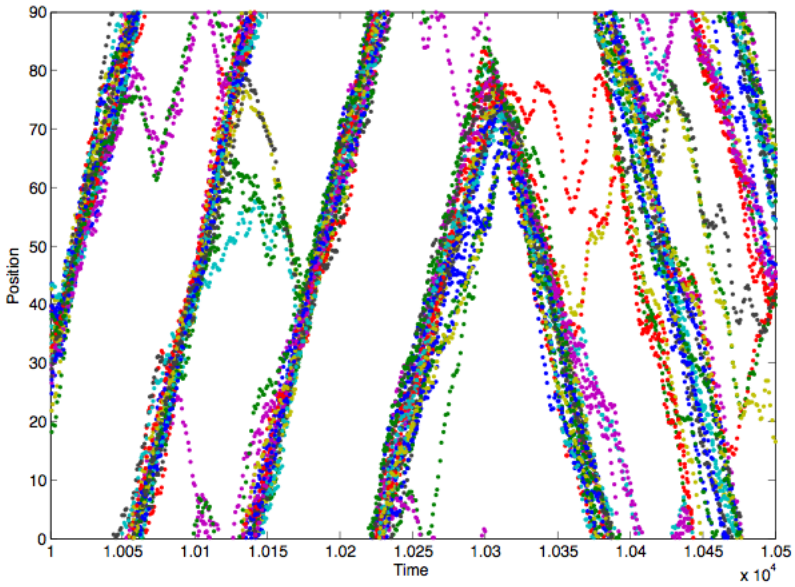


Figure 2.1: An example swarm produced with the SPP model.

These rules were put together and implemented in *SppModel.m* (**Appendix A**). **Fig. 2.1** shows the positions of each locust in the swarm from $t = 10000$ to $t = 10500$. In this simulation there are 30 locusts simulated for 15000 time steps with Δt set to 1, giving a total time T of 15000. The interaction radius is set to 4 and the length of the domain is 90. Locusts are initialized with random positions and random velocities.

It can be seen in **Fig. 2.1** that the locusts align and move as a swarm for some time before suddenly changing directions. A way to quantify this behavior is to find the average velocity of all the locusts for each time step. This is called the alignment, calculated as

$$X(t) = \frac{1}{N}\sum_{i=1}^{N} u_i(t) \tag{2.6}$$

Alignments around ±1 reflect collective swarming behavior, while an alignment around 0 reflects random unaligned behavior. The alignment for the simulation in **Fig. 2.1** is plotted in **Fig. 2.2**.

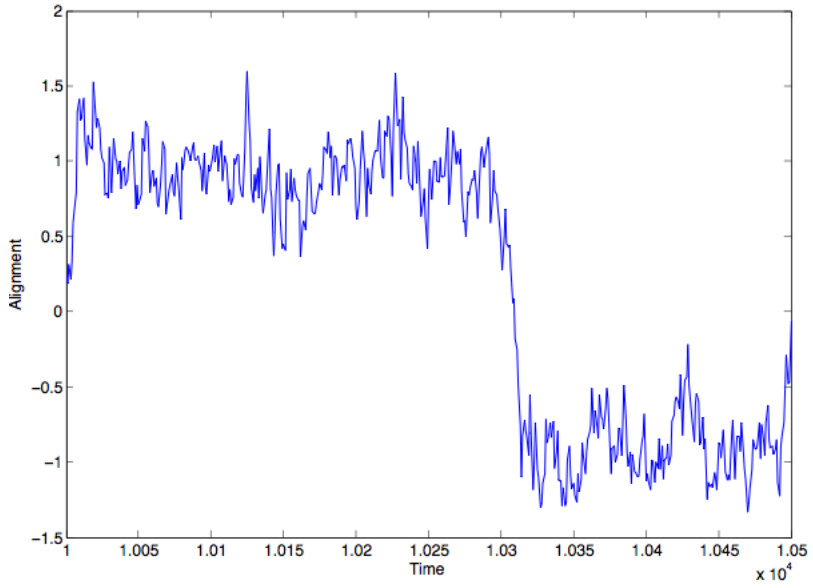


Figure 2.2: Alignment of the entire simulation from Fig. 2.1. The alignment variable quickly reaches +1 and switches rapidly at $t = 10300$ to -1.

Comparing **Figs. 2.1** and **2.2,** the relation between the swarm and the calculated alignment becomes apparent. When the swarm is moving in one group with a positive velocity around time t = 500, the alignment stays around +1. At time t = 650, when the swarm abruptly changes direction, the alignment passes through 0 and settles at -1 once the swarm starts moving together in the opposite direction. This switching behavior persists throughout the simulation.

In this paper, the same parameters as in Yates and Buhl *et al*. are chosen [14, 15]; the time step Δt is set to 1, the length of the domain is set to 90, and the interaction radius is set to 4. The number of locusts is set to 30. To create a large enough data set for data mining, a simulation was run for 50,000 time steps. Each time step contains data on the position and velocities of all 30 locusts, creating a 60-dimensional data set with 50,000 data points. This constitutes the data set from which all following analysis will take place.

2.2 Defining Stable Regions and Switch Points

To discover any potential hidden structures within the simulation, it is necessary to identify which parts represent "stable regions," when the alignment stays near ±1, and which parts represent "switch regions," when the alignment is switching between the two stable states. To define these states by simply checking if the alignment at each step were above some threshold would not work because the fluctuations within stable states that sometimes cross zero would be erroneously marked as switch points. To prevent this, switch points were only marked when the alignment had an opposite sign for 40 consecutive steps. Stepping backwards from this point, for positive alignments, the first point that went below -0.5 was marked as the "end switch point," and the first point that

went below 0 was marked as the "start switch point." For negative alignments, the thresholds were above +0.5 and above 0. This method was implemented in *switchPoints.m* in **Appendix B**.

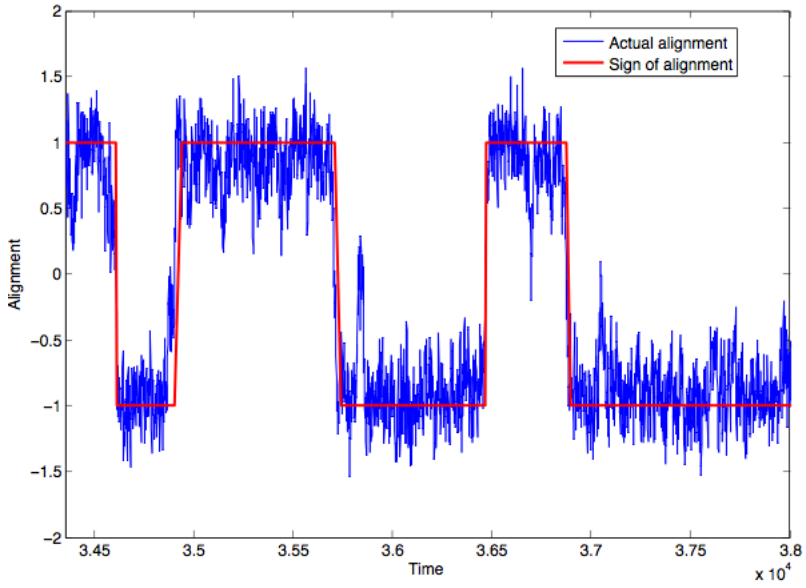


Figure. 2.3: The actual alignment of part of the simulation and the detected switch points.

Using this algorithm, the region between times $t = i$ to j is defined as a "stable region" if there are not 40 consecutive points between i and $j + 40$, with $j > i + 40$, with an alignment that has the opposite sign of the alignment at time $t = i$. This is condition is represented mathematically in Equation 2.7.

$$\left(\sum_{k=t}^{t+40} a_i \,\mathrm{sign}(a_k)\right) > -40, \text{ for all } t \in i \le t < j \tag{2.7}$$

In Equation 2.7, a_i is the alignment at time $t = i$, and is therefore the alignment of the results stable region. The alignment at time $t = k$ is given by a_k. The function sign(x) returns 1 if $x > 0$, 0 if $x = 0$, and -1 if $x < 0$.

The switch points and the alignment for a selection of the simulation are shown in **Fig. 2.3**. The spike near time t = 35800 is correctly not counted as a switch point. The

switch region at time t = 34901 finishes in 38 time steps, while the switch region at time t = 36463 lasts for 10 time steps.

The random term in the velocity calculation in Equation 2.2 is large relative to the range of velocities in the population, so for some lag time τ, the state (velocities of all the locusts) at time $t + \tau$ will not depend on the state at time t. Finding the dependence of the current state on previous states can show how deterministically the swarm behaves. Autocorrelation measures this value. It is the time ensemble average of the product of the normalized value at time t and time $t + \tau$. See **Appendix C** for details about the calculation. For a uniformly random sample, the autocorrelation is very low for all lags τ. For a periodic signal, the autocorrelation peaks at lags that are equal to multiples of the period.

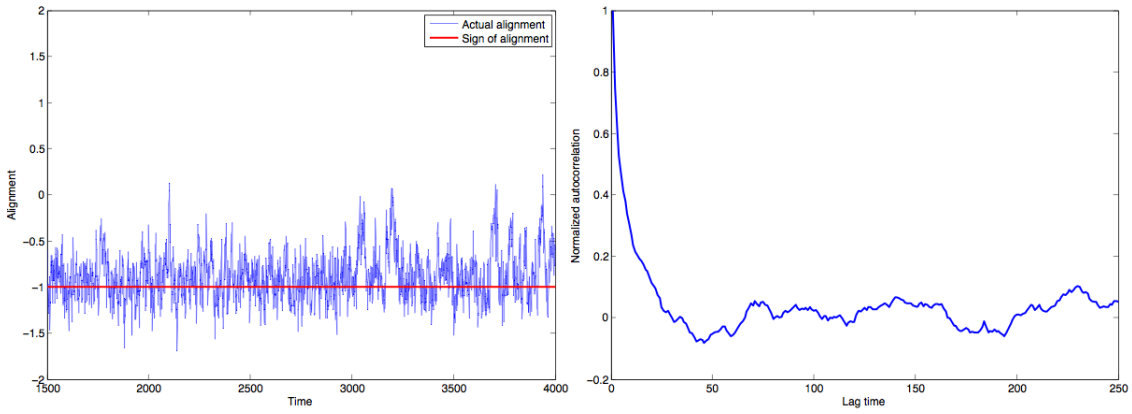


Figure. 2.4: Left: Alignment for a stable region in the simulation. Right: Autocorrelation of the data in the stable region.

The autocorrelation for the stable region from times t = 1500 to 4000 is shown in **Fig. 2.4**. The autocorrelation is normalized so that for a lag of 0, the autocorrelation is 1. The value sharply drops off and crosses 0.10 at a lag of 21. For larger lag values, the autocorrelation fluctuates about 0 with an amplitude of 0.10. This indicates that within a stable region, the state at time t has a negligible impact on the state at time $t + \tau$. In other

words, the system only has a “memory” of 21 time steps, after which the random term erases previous information. To make sure that the "memory" of a switch region did not affect the stable regions that were studied, the points between 40 steps after the end switch point and 40 steps before the start switch point of the next switch region were considered as stable regions. In all following work, only stable regions lasting longer than 450 steps are studied.

Because there are not enough points for an accurate autocorrelation calculation in the switch regions, an alternative method of classification was used. To make sure the start switch points were preceded by a long enough stable region, only start switch points with a stable region lasting longer than 450 time steps were studied. This gave a sample of 32 start switch points. From each of the 40 steps before each studied start switch point and including the start switch point, the simulation was rerun with the same initial positions and velocities for 40 trials for a time length up to the time of the start switch point plus 25. This was implemented in *switchFrequency.m* in **Appendix D**.

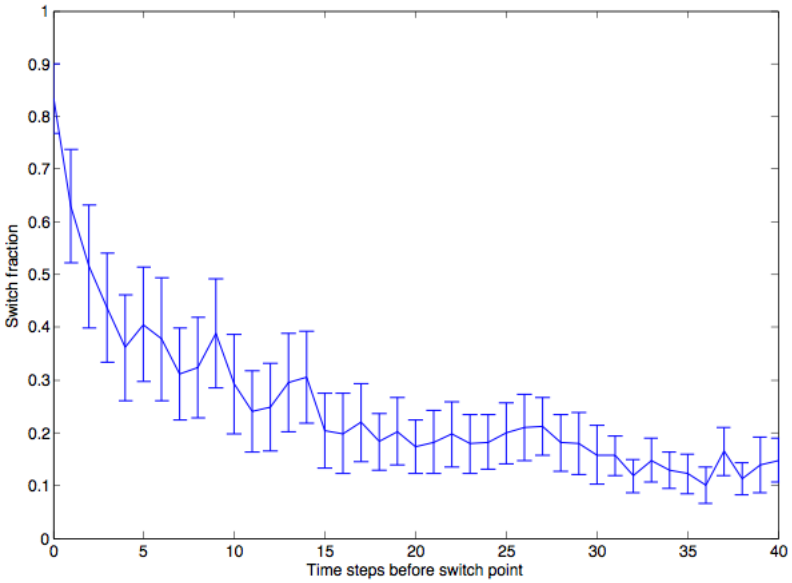


Figure 2.5: Average switch fraction over all 32 studied start switch points. Error bars represent a 95% confidence interval.

The switch fraction, the fraction of trials that resulted in the alignment changing signs by the end of the short simulation, averaged over all start switch points was plotted as a function of time steps back from the start switch point in **Fig. 2.5**. About 80% of the time, a simulation initialized with the same state (positions and velocities of all locusts) as a start switch point will indeed change its alignment. The switch fraction quickly drops off, decaying roughly exponentially to 0.20 at 15 steps before the start switch point.

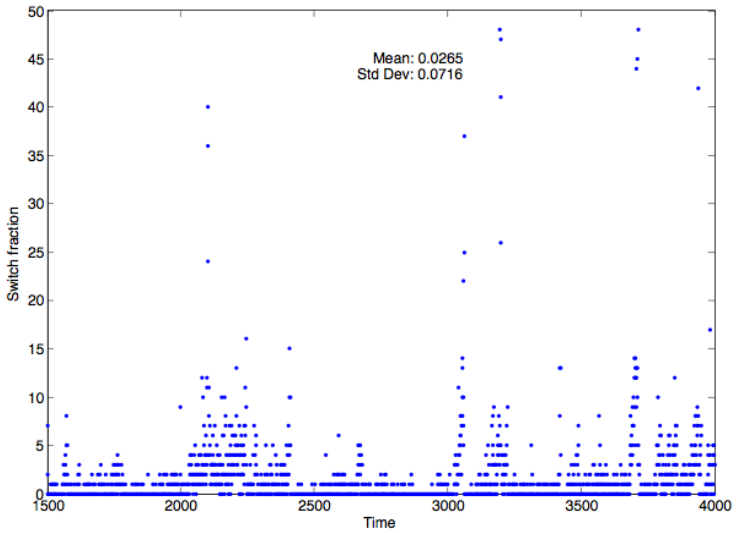


Figure. 2.6: The switch fraction at each time step inside a stable region. The average switch fraction is 0.0234, so it is very unlikely that a state selected inside of a stable region will switch in 20 steps.

To better appreciate these values, for each time step in the stable region from t = 1500 to 4000, the simulation was run with the given initial state at that time step for 20 steps over 50 trials. **Fig. 2.6** shows that the switch fraction rarely goes above 0.10, and the average value is 0.0234.

Each start switch point was classified based on the behavior of its switch fraction. When the switch fraction dropped to 0.20 after 6 steps back from the start switch point and only went above that value less than three times, it was classified as a "low switch." If it has a switch fraction above 0.40 after 10 steps back and a switch fraction above 0.24

after 25 steps back, it was classified as a "high switch." If its behavior fell between these two descriptions and it did not exhibit any large fluctuations above 0.40 between 10 and 25 steps back, it was classified as an "exponential switch" because of its appearance. If the behavior of the start switch point did not fall into any of these three categories, that start switch point was not classified and only analyzed when all the start switch points were analyzed together. In total there were 8 low switches, 7 high switches, 6 exponential switches, and 11 unclassified start switch points.

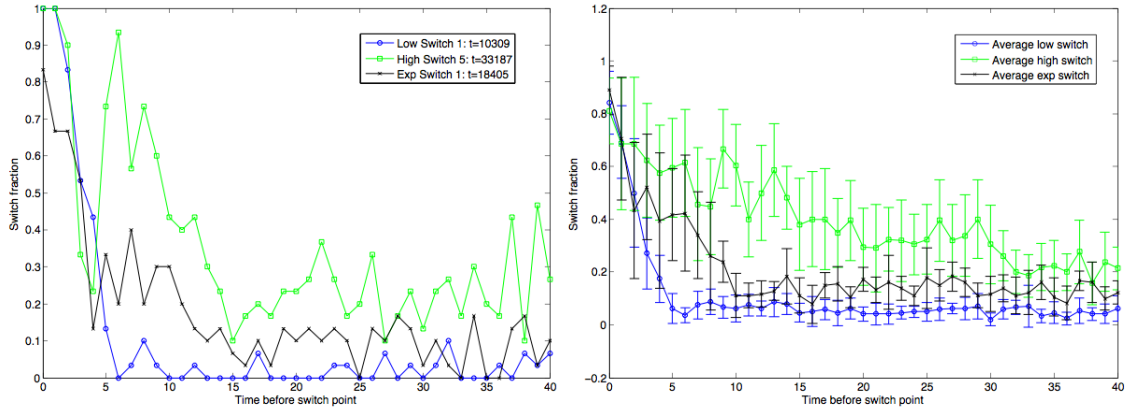


Figure 2.7: Left: An example of the switch fraction behavior of each type of start switch point. Right: The switch fraction average of each type of switch. Error bars represent a 95% confidence interval.

The behavior of the switch fraction for each classification of start switch point is shown in **Fig. 2.7**. Now we have determined the three types of start switch points and the stable regions. Before analyzing the data for patterns, it is useful to recognize the symmetry of the system. The direction of positive velocity is arbitrarily chosen to be from left to right. Changing the definition from right to left would not change the behavior of the SPP model. Therefore, values such as the standard deviation of velocities of the locusts would not change as well. In order to make comparisons of statistical data easier, when studying stable regions or start switch points with negative alignment, all velocity data was multiplied by negative 1, while for all positions, modulo 90 of the

product of the position and negative one was taken. This ensures that the swarm is "facing" in the correct direction.

2.3 Basic Statistical Analysis

For each category of start switch points with the 40 steps before them, the mean, standard deviation, skewness, and kurtosis of the velocity, and the standard deviation, skewness, and kurtosis of the position were calculated. The same was done for 41-step stable regions taken from each region lasting longer than 450 steps. In the simulation, 8 low switches, 7 high switches, 6 exponential switches, 11 unclassified switches, and 33 stable regions were identified and studied. While the mean and standard deviation are well known measures, the skewness and kurtosis require explanation.

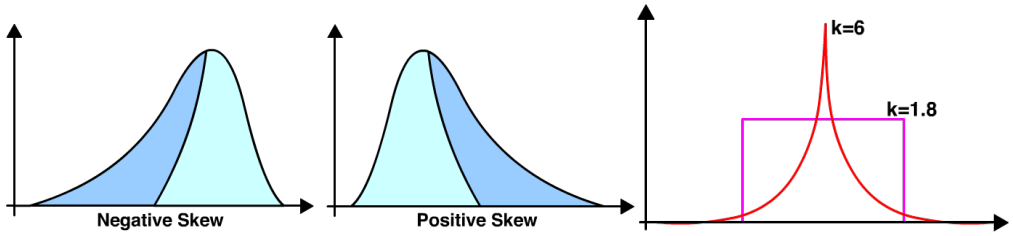


Figure. 2.8: Left: A negative skewness indicates that most of the data is to the right and there is a long tail to the right. A positive skewness indicates the opposite. Right: The red peak has a kurtosis of 6, indicating that most of the data is in the peak and the tails, while the purple uniform distribution has a kurtosis of 1.8, indicating a lack of a peak or tails.

The skewness is the third standardized moment, that is, the ratio of the third central moment and the third power of the standard deviation. It is a measure of how much of the data falls to one side of the mean. A negative skew indicates a long left tail with most of the mass concentrated on the right side of the graph, while a positive skew indicates the opposite (see **Fig. 2.8**). Kurtosis is the fourth standardized moment of the data. While there are several interpretations of this quantity, it is easiest to think of it as a measure of peakedness, or a measure of how much data falls into the peak and into the

tails as opposed to the areas in between. The Bernoulli distribution with p = 1/2 has the lowest kurtosis of 1 because none of the data is in a central peak or a long tail. The normal distribution has a higher kurtosis of 3 because it has more peak character (see **Fig. 2.8**).

Sometimes the positions of locusts are around the 'ends' of the domain at 0 and 90. This creates an artificially large standard deviation when in fact the locusts are close to each other. To fix this, the positions were shifted such that the center of mass of the swarm was near 45 at the time of the start switch point. This made the statistics on the position data much more accurate. With all the position data shifted, statistical measures can now be taken.

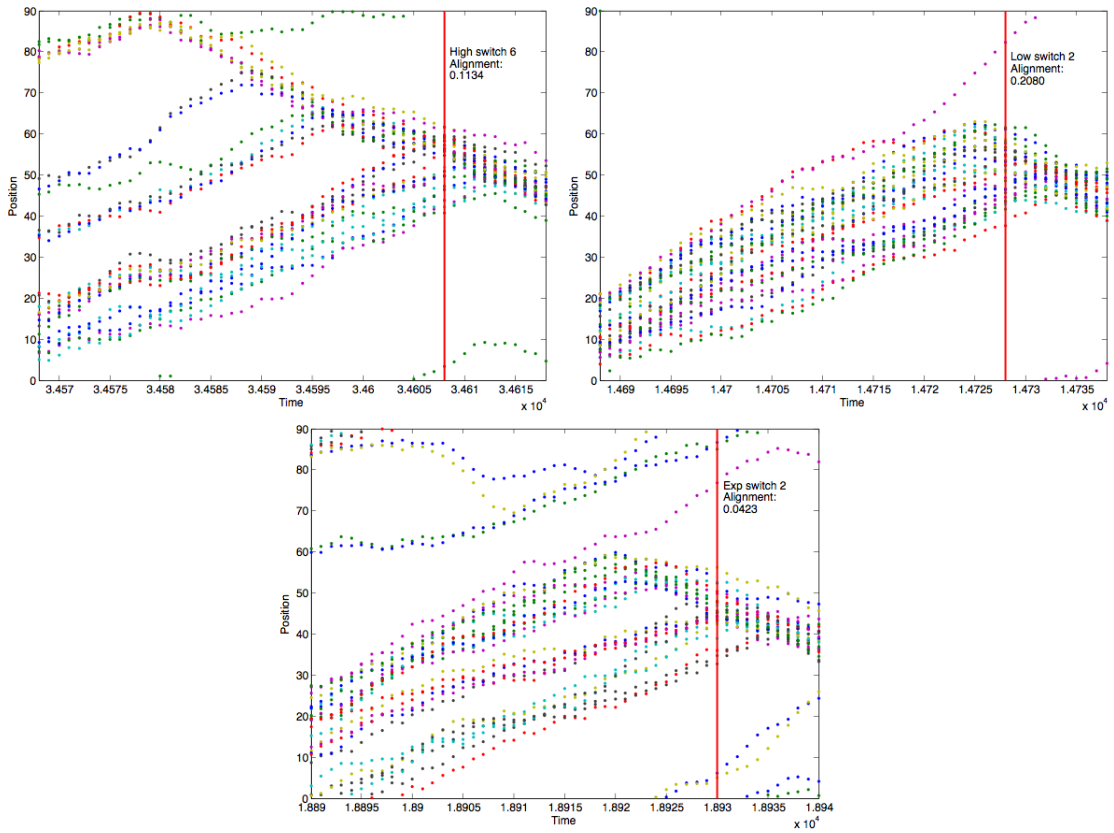


Figure. 2.9: Top left: Typical swarm positions for a high switch, with two separate swarms colliding. Top right: Typical swarm positions for a low switch, with the swarm seeming to change direction all at once. Bottom: A swarm from one of the exponential switches, exhibiting behavior in between that of a high switch and a low switch. All start switch points are marked with a red vertical line.

Looking at the positions of the locusts leading up to each start switch point, a few qualitative points can be made. High switches almost always consist of two equally sized swarms crashing into each other, while low switches have one very small and one very large swarm. There are also low switches in which the entire swarm spontaneously changes direction. The exponential switches consist of both types of scenarios, as well as more chaotic states. Representative snapshots are given in **Fig. 2.9**.

Table 2.1: Statistical measures for stable regions

Velocity	Average Value	Standard Deviation
Mean	0.9163	0.2157
Standard Deviation	0.9594	0.1372
Skewness	-0.2030	0.4206
Kurtosis	2.4439	0.8973
Position		
Standard Deviation	18.8965	8.2042
Skewness	0.7875	1.0005
Kurtosis	4.0491	2.8992

These values stay relatively flat inside the stable regions. They are summarized in Table 2.1. See **Appendix E** for the corresponding graphs. Using these values as a baseline, we can look at the corresponding measures on the switch regions.

For the switch regions, the average velocity (alignment) increases from near 0.1 to near 0.9 as the simulation moves farther back from the start switch point. This is unsurprising because the start switch point has an alignment around 0 by definition. The high switches tend to have a lower average velocity than the other switches at corresponding times. The standard deviation of the velocity follows the opposite trend: at the start switch point it averages around 1.25 and 40 steps before the switch point it averages around 1. This is also unsurprising because if the alignment is near 0, the velocities are bound to be more spread out than if they were at a stable alignment. Here, the high switches tend to have a higher standard deviation than the other switches. The

velocity skewness appears to also decrease slightly further from the start switch point. At the start switch point, the average skewness is +0.05, but it is negative for a majority of the time around the same value (-0.2) as the velocity skewness for the stable regions. A negative skew indicates a long left tail. The distributions of velocities tend to be such that the velocities cluster around the alignment with the major outliers being negative. There does not appear to be any difference between the different switch types. Finally, the kurtosis of the velocity for the switch points does not appear to differ significantly from that of the stable region.

The standard deviation of the position represents the spread of the locust swarm. Moving closer to the start switch point, the standard deviation decreases down to around 10 (see **Fig. 2.10**). One common cause of a switch is a locust at the edge of a diffuse swarm spontaneously changing direction. As it moves back through the diffuse swarm, it tightens the swarm and changes the alignment of its neighbors. This is reflected in the shrinking standard deviation. Szeto (2009) observed this behavior when a "leader locust" biased with a positive velocity was launched into a swarm with a negative alignment [27]. The standard deviation tends to be higher for high switches than for low switches.

The skewness of the position increases slightly with the time before the start switch points; however, the value can be artificially inflated if some locusts have looped around the domain, giving values like 0 or 90. The kurtosis of the position at the start switch point is greater than that of a stable region. It decreases up to 12 steps before the switch point, then increases to near the baseline. The exceptions are the high switches: the kurtosis continues decreasing to around 1.5. Such a low kurtosis could indicate that two separate swarms exist. Indeed, the high switches tend to look like **Fig. 2.9** (top left).

However, this is also due to how the position data was centered. The data at the switch point was centered at the middle of the axis near 45, and coincidentally, it takes about 40 steps for two swarms starting at 89 and 1 to collide.

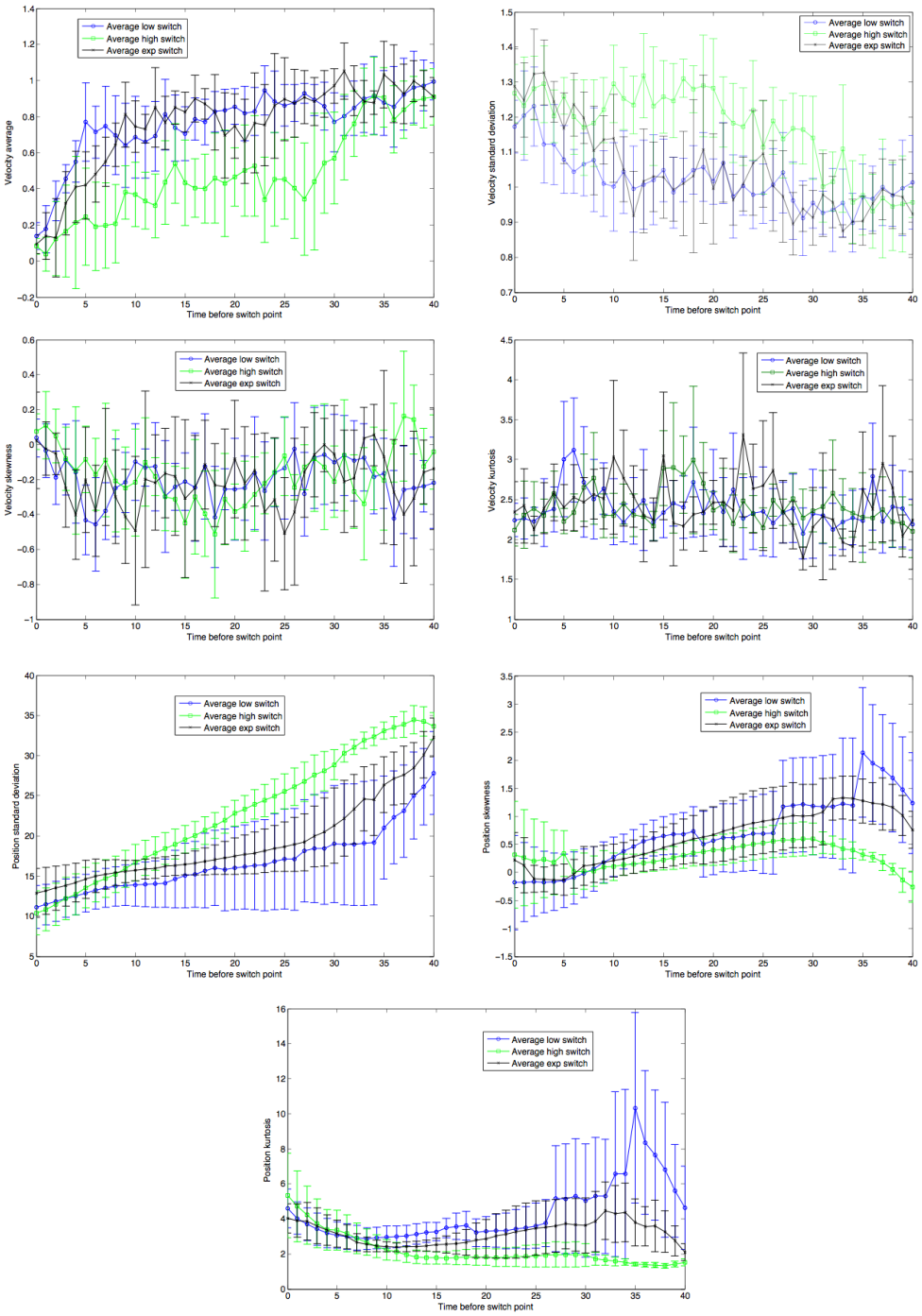


Figure 2.10: The average trends of various statistical measures as they near each type of switch. The label on the y-axis indicates which variable is being measured. Error bars represent a 95% confidence interval.

The switch percent for each switch was plotted as a function of each statistical value to check for any correlation. These plots can be found in **Appendix F**. Both the coefficient of determination (R^2) from simple linear regression and the Pearson product-moment correlation coefficient (PCC) were calculated for each figure (see **Appendix G** for PCC calculation). The R^2 values range from 0 to 1, with 0 indicating no trend and 1 indicating a perfect fit of the regression line. The PCC value ranges from -1 to +1 and indicates how much two variables linearly depend on each other. Each measured statistic had an R^2 value greater than 0.96, however, this is not a very good indicator of dependence. Most of the switch fraction values are low, between 0 and 0.2, so the linear fit is always heavily biased to have a slope near zero. The PCC value gives more information. It has a value near 0 for all measurements except for position standard deviation and position kurtosis. For the standard deviation, the PCC is 0.5060 (see **Fig. 2.11**), indicating a good linear fit with switch percentage. This backs up the data from the plot of standard deviation against time. For the kurtosis, the PCC is -0.1720. This negative trend matches the claims made in the above section.

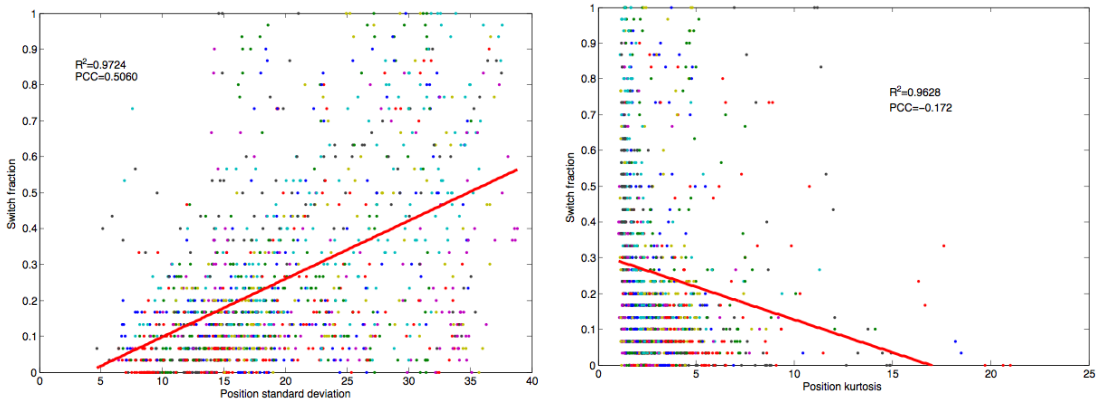


Figure 2.11: Left: There is a rough positive correlation between the position standard deviation and the switch fraction. Right: the position kurtosis has a rough negative correlation with the switch fraction.

The above analysis indicates that important measurements are the average velocity (alignment), the standard deviation of the velocity, the standard deviation of the position, and (to a lesser extent), the kurtosis of the position. These measurements should be kept in mind when analyzing the model because they are easy to measure and provide a first-order picture of the swarm.

3 Dimensional Reduction Results

The following sections contain the results of two types of dimensional reduction analysis. PCA, a linear dimensional reduction technique, is used to see if there exists a new set of orthogonal variables that can explain the variance in the data better than the original variables. Then diffusion map analysis (DMA) is used to see if a lower dimensional manifold can represent the 60-dimensional data and to discover any high priority variables. Finally, we investigate the effects of manipulating individual locusts in order to induce a change in alignment.

3.1 Principal Component Analysis

Principle component analysis is a linear transformation of a set of data into a set of linearly uncorrelated variables known as principle components. The following description and detailed calculations are given in Chatterjee (2000) [20]. Consider taking N measurements of m state variables (for example, coming from m velocity probes in a fluid or m positions of individuals in a swarm taken over N time steps). The data is arranged into the N x m matrix A such that the element A_{ij} is the measurement from the j^{th} probe taken at the i^{th} instant. The main process of PCA is the singular value decomposition (SVD) of the data matrix A into the form $U\Sigma V^{\text{T}}$, where U is an N x N

orthogonal matrix, V is an m x m orthogonal matrix, and Σ is an N x m matrix only populated along the diagonal. The values of Σ, known as the singular values, are in decreasing order and are denoted σ_i. The end result of PCA is a new set of variables that account for the largest amount of variance in a given direction.

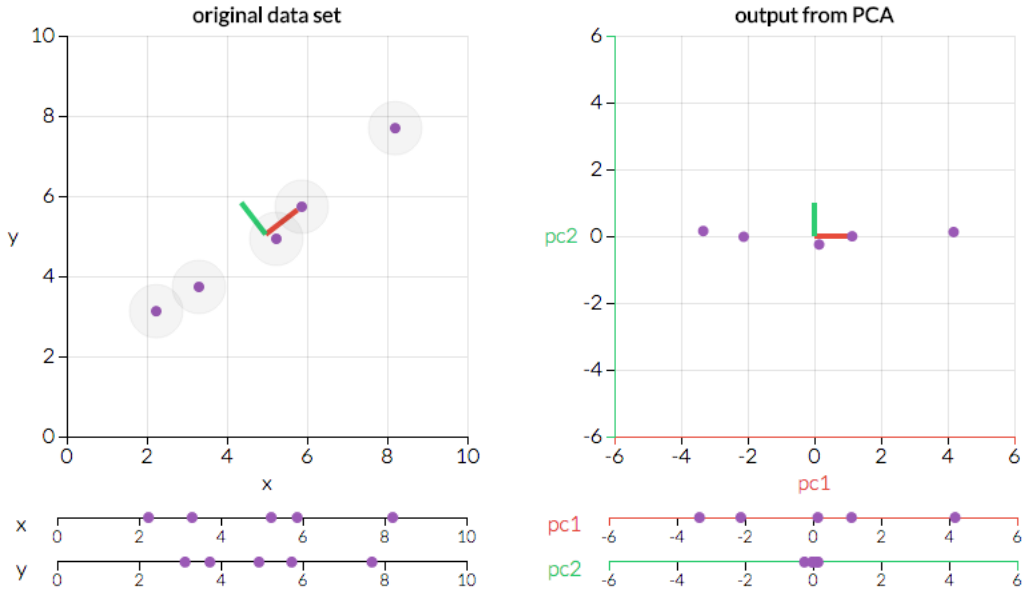


Figure 3.1: PCA analysis on a data set (left) reveals most of the variance is described by one new direction, or principal component. Representing the two-dimensional data along the one-dimensional component pc1 would result in only a small loss of information. Figure from Powell & Lehe (2015) [28].

A useful geometric interpretation of PCA is to imagine A as a linear operator that maps a vector from m-dimensional space to N-dimensional space. If the unit sphere in m-dimensional space were mapped to N-dimensional space, it would be mapped to an ellipsoid. The singular values σ_i are the lengths of the principle radii of the ellipsoid, and the columns of U give the directions of the principle radii. The columns of V are the pre-images of the principle radii. The goal of PCA is to create a lower rank approximation in k dimensions of A called A_k. The matrices U_k and V_k are created by taking the first k columns of U and K, and Σ_k is created by taking the leading k x k principal minor submatrix. A_k is then given by $U_k\Sigma_k V_k^{\mathrm{T}}$. The magnitude of the singular values σ_i represents the contribution of each corresponding variable. **Fig. 3.1** shows PCA being

applied to reduce a two-dimensional data set into a one-dimensional subspace. In this example, σ_1 is much larger than σ_2. A three-dimensional example is given in **Fig. 3.2**.

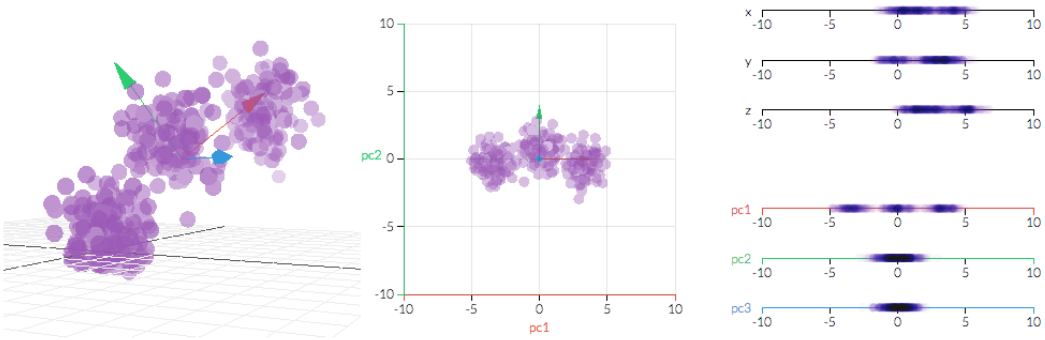


Figure 3.2: Left: The original three-dimensional data with the principal components shown as arrows. Center: The data represented in two-dimensions using the first two principal components. The third component points into the page. Right: The third component has the least variance, so only a small amount of information is lost in the two-dimensional representation. Figure from Powell & Lehe (2015) [28].

The percentage contribution of σ_i is simply σ_i divided by the sum of all singular values. PCA works well then the first two or three singular values represent 90% of the variance. For example, in **Fig. 3.1**, σ_1 accounts for 99.8% of the variance, indicating that the one-dimensional representation is useful.

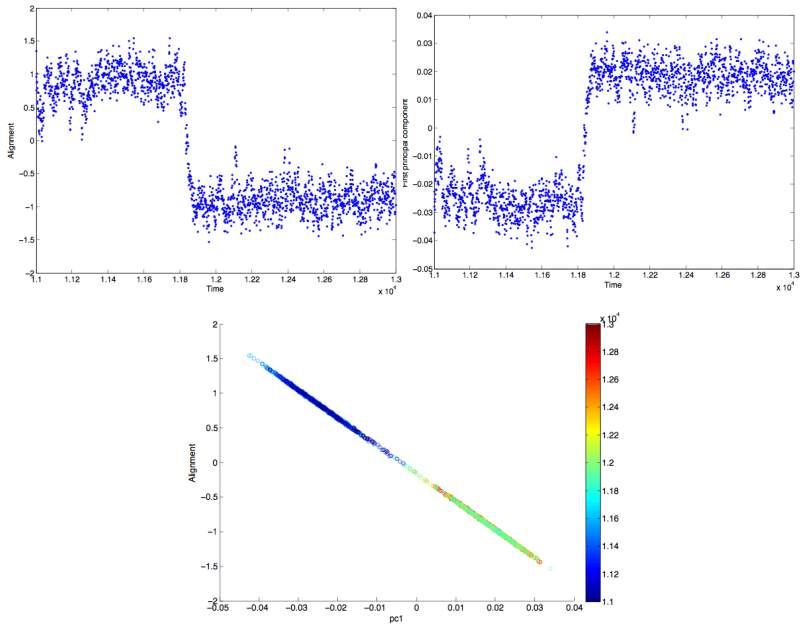


Figure. 3.3: Top left: The first principal component from times t = 11000 to 13000. Top right: The alignment over the same time period. Bottom: The first principal component and alignment correlate very strongly. The color bar represents the time of the simulation.

PCA was carried out on times t = 11000 to 13000 because it contained one start switch point. A simplified implementation is available in **Appendix H** as *Pca.m*. Plots of the alignment, the second eigenvector, and the correlation between the two are in **Fig. 3.3**. The strong correlation indicates that the alignment variable represents an important feature of the swarm.

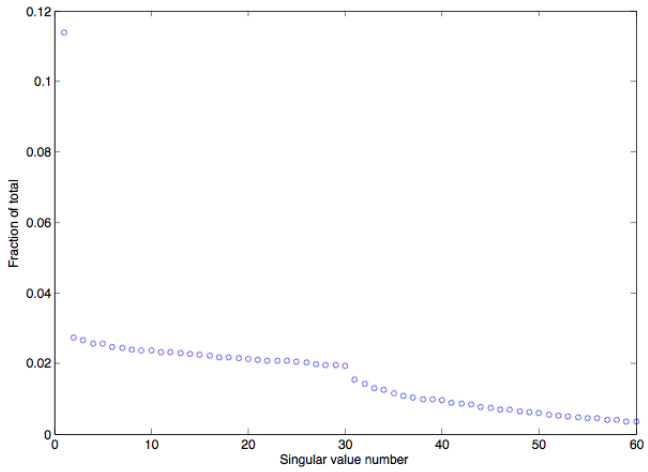


Figure 3.4: The percent contributions of the singular values of the PCA performed in Fig. 3.3. The first value is much greater than the rest of the singular values, but still accounts for less than 12% of the variation.

The singular values contributions for the PCA analysis are given in **Fig. 3.4**. There is a large gap between the first and second singular value, indicating that the variance of the data is largely represented along the first principal component axis. Still, this only represents about 11.4% of the variance of the data, showing that while the first principal component is much more important the following components, it does not account for a majority of the variance in the overall dataset. The one-, two-, and three-dimensional embeddings are shown in **Fig. 3.5**. The first principal component, corresponding to the alignment, is shown to be the primary axis between two clusters of data.

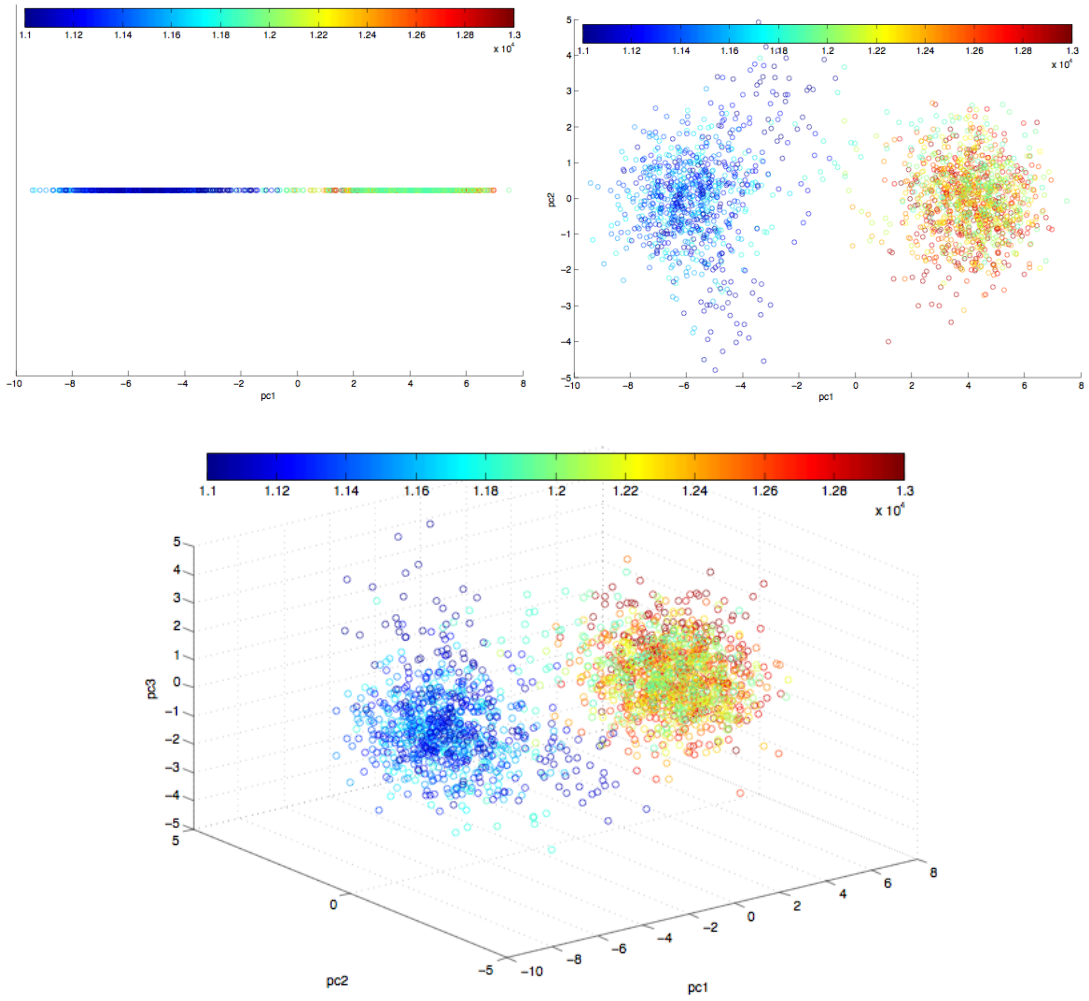


Figure 3.5: One-dimensional (top left), two-dimensional (top right) and three-dimensional (bottom) embeddings of the original data set. The first principal component sets up the structure for the clustering evident in the two- and three-dimensional representations. The color bar represents the time of the simulation.

To see if there was some difference between the PCA results of stable regions and the different switch points, PCA was also carried out on each of the studied stable regions and switch types.

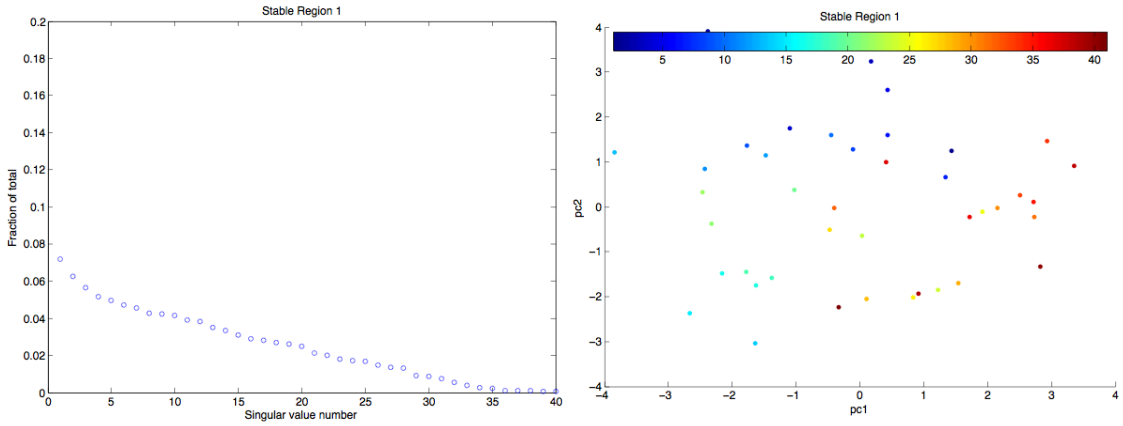


Figure 3.6: PCA on the first stable region. Left: The relative contribution of the first singular value is even less than in Fig. 3.4. Right: The two-dimensional embedding of the data. The color bar represents the time of the simulation.

Because there is no change in alignment, the overall variance of the data is significantly reduced. This is reflected in the magnitude of the singular values. The first singular value in **Fig. 3.6** only accounts for 7.2% of the total variance in the first stable region. The embedding in two dimensions is given as well. For all stable regions studied, a similar pattern was observed.

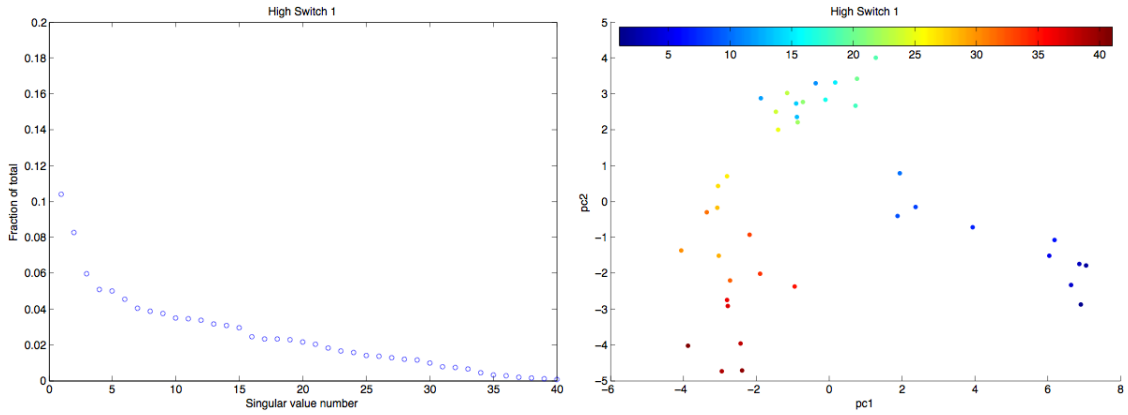


Figure 3.7: PCA on the first high switch. Left: The relative value of the first singular value is greater than for the stable region, but still less than the sample in Fig. 3.4. Right: The two-dimensional embedding of the data. The color bar represents the time of the simulation.

The singular value contributions and two-dimensional embedding of the first high switch are given in **Fig. 3.7**. As with the stable region, the first singular value does not represent a majority contribution of the variance. All switches followed a similar pattern.

This indicates that there is no linear substructure within the 60-dimensional data set that accompanies a switch in alignment.

The next section focused on applying the nonlinear DMA to the dataset to discover underlying lower-dimensional structures.

3.2 Diffusion Map Analysis

Diffusion map analysis is a nonlinear dimensional reduction technique that maps the "diffusion distance" between points in *n*-dimensional space to a lower dimensional Euclidean space. The complex geometric structures in *n*-dimensional space may be revealed to have more efficient lower dimensional manifolds. This is visualized in **Fig. 3.8**.

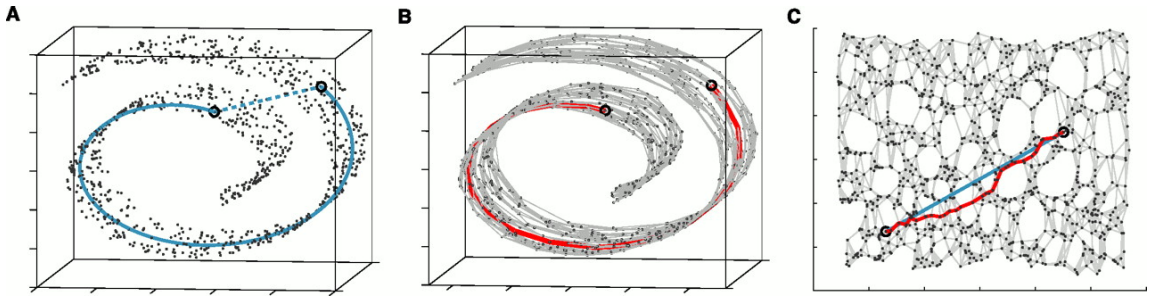


Figure 3.8: A. The "Swiss roll" data set is presented. The blue dotted line represents a shorter Euclidean route between the two marked points, but it would not reveal the underlying two-dimensional structure of the data seen along the blue solid line. B. The shortest diffusion distance connecting the two points by traversing other points in the data set. C. The two-dimensional embedding is such that the diffusion distance in the original set (red line) and the Euclidean distance in the embedding correspond. Figure from Tenenbaum *et al*. (2000) [29].

The DMA algorithm is laid out by de la Porte *et al*. (2008) and summarized in four basic steps [30]. First the connectivity of a given high dimensional data set of N points is determined. The connectivity between x_i and x_j is defined as the probability from jumping from x_i to x_j in one step of a random walk. It is proportional to some likelihood function, k, called the diffusion kernel. The Gaussian kernel, given in Equation 3.1, is a popular choice. The kernel matrix K is then created such that $K_{ij} = k(x_i, x_j)$.

$$k(x_i, x_j) = \exp\left(\frac{-\left|x_i - x_j\right|^2}{\varepsilon}\right) \tag{3.1}$$

Second, the rows of the kernel matrix K are normalized to create the diffusion matrix P. Each entry P_{ij} represents the probability of a single step between x_i and x_j. When P is raised to the power t, the entries represent the probability of taking a journey t steps long between two points.

Third, the diffusion distance is used to calculate the eigenvectors and eigenvalues of the diffusion matrix P. The diffusion distance $D_t(x_i, x_j)^2$, quantifies the similarity of two points as a function of their connectivity at time t, and is defined below as

$$D_t^2(x_i, x_j) = \sum_{k=1}^{N} \frac{\left|P_{ik}^t - P_{jk}^t\right|^2}{D_{kk}} \tag{3.2}$$

where D_{kk} is the degree of vertex k. The eigenvectors ψ_k and eigenvalues λ_k are related to the diffusion distance by equation 3.3. It is proved by de la Porte *et al*. (2008) [30].

$$D_t^2(x_i, x_j) = \sum_{k=1}^{N} \lambda_k^{2t} \left(\psi_k(i) - \psi_k(j)\right)^2 \tag{3.3}$$

The eigenvalues are arranged in descending order, and the first eigenvalue λ_1 and eigenvector ψ_1 are discarded as trivial. By using the next m dominant eigenvalues and eigenvectors, the data can be mapped onto the m-dimensional subspace.

This is the fourth and last step of the DMA algorithm. The m-dimensional embedding is given by the following transformation $\Psi_t(x)$ below.

$$\Psi_t(x) = (\lambda_2^t \psi_2(x), \lambda_3^t \psi_3(x), ..., \lambda_m^t \psi_m(x)) \tag{3.4}$$

Often t is taken to be 0 so that the transformation simplifies to

$$\Psi(x) = (\psi_2(x), \psi_3(x), ..., \psi_m(x)) \tag{3.5}$$

$\Psi(x)$ in Equation 3.5 is used for the DMA embedding in the following analysis.

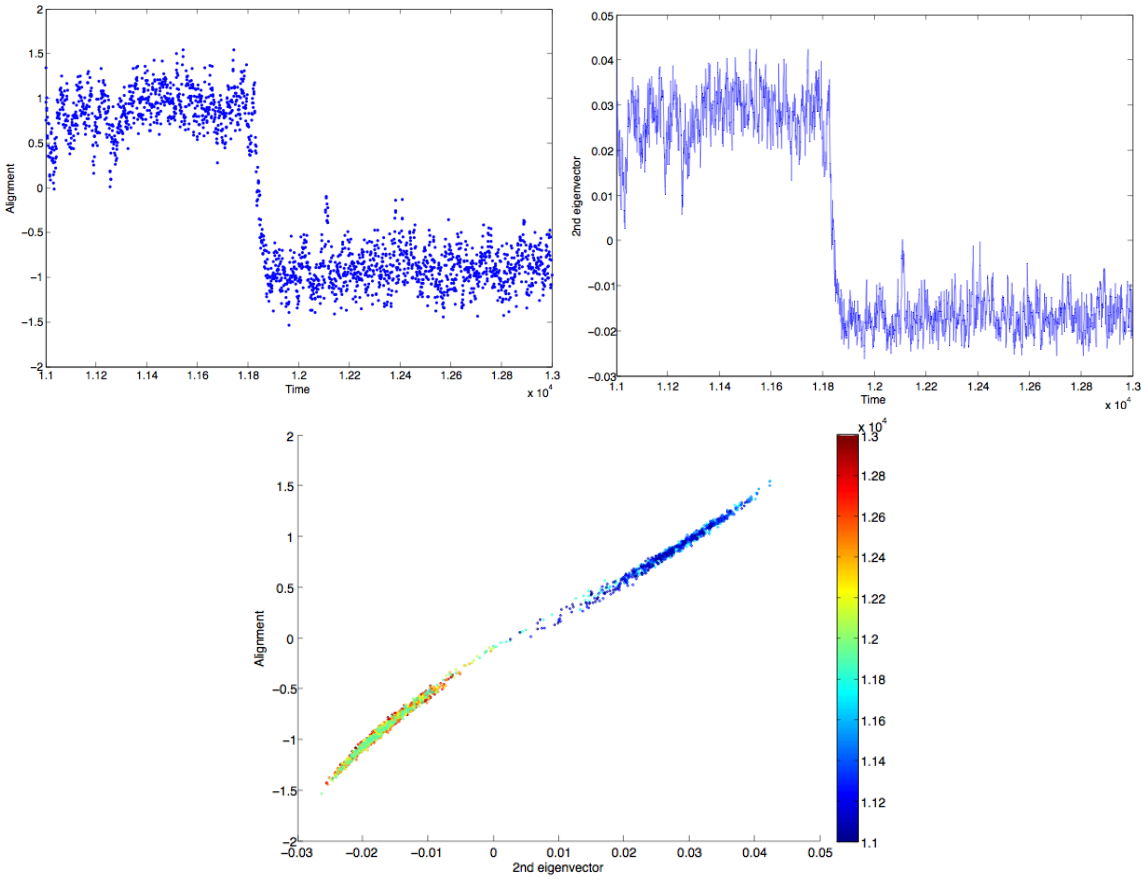


Figure. 3.9: Top left: The second eigenvector from times t = 11000 to 13000. Top right: The alignment over the same time period. Bottom: The second eigenvector and alignment correlate very strongly. The color bar represents the time of the simulation.

Wilkes (2013) showed that the second eigenvector corresponds to the alignment, indicating that the alignment is a high priority variable for the system [31]. DMA was performed on all data from times t = 11000 to 13000, the same region as the PCA analysis, because it contained one start switch point. The Matlab code *diffusionMap.m* is given in **Appendix I**. Plots of the alignment, the second eigenvector, and the correlation between the two are in **Fig. 3.9**. The strong correlation indicates that the alignment variable gives information about the structure of the swarm.

The third, fourth and fifth eigenvectors were plotted against each of the statistical measures and the critical number and switch fractions (discussed in Chapter 4). See

Appendix J for an example of the third eigenvector plotted several of these values. No further correlations between eigenvectors or other values were observed.

Wilkes (2013) has shown that performing equation free analysis on the second eigenvector indeed recreates the same potential well as the original simulation [31]. This means that instead of saving all the individual locations and velocities of the locusts (60 variables), just saving the second eigenvector (1 variable) could provide an accurate lower dimensional picture of the data insofar as the alignment is important. A rough recovery of the simulation would require scaling the second eigenvector to the proper range of the alignment, and then initializing locusts such that they have the given alignment [31].

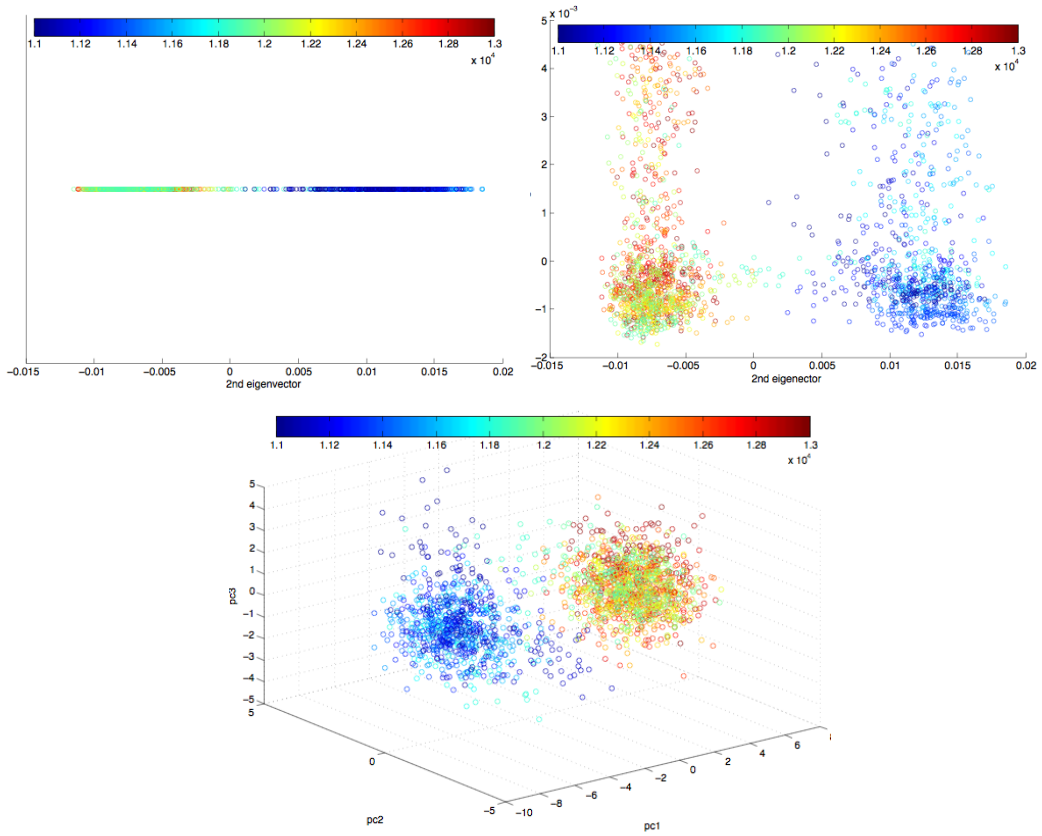


Figure 3.10: Data embedding in the first (top left), second (top right) and third (bottom) eigenspaces resulting from DMA on the data. Clustering is again evident as in Fig. 3.5, but it more pronounced. The color bar represents the time of the simulation.

Embedding the data into the one-, two- and three-dimensional eigenspaces shows that the second eigenvector, which corresponds to the alignment, indeed organizes the data into groups (see **Fig. 3.10**). The lower dimension structure of two clusters becomes apparent especially in the three-dimensional embedding, where the two clusters are clearly separated.

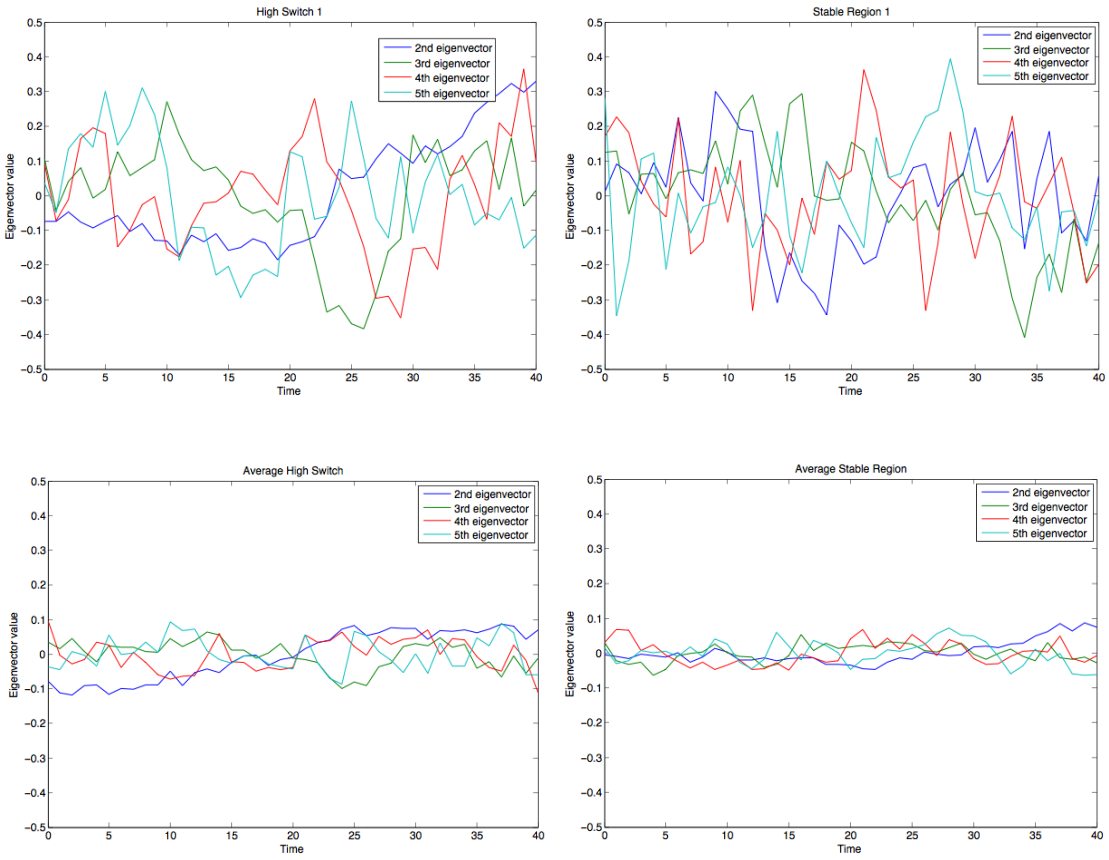


Figure 3.11. Top left: DMA eigenvectors for high switch number 1. Top right: The same for a stable state. Bottom left: Average over all switches. Bottom right: Average over all stable states.

Finally, diffusion maps for each of the switches types studied and the stable regions were created, and the second through fifth eigenvectors were plotted over time to see if there was a difference between the stable regions and the switch regions. No new trends were identified. An example of the eigenvectors for the diffusion map from one of the high switches and from one of the stable states is in **Fig. 3.11**. Also plotted are the

averages of all the eigenvectors for all switches and all stable regions. There is no significant difference between the graphs, indicating that the eigenvectors do not vary in form between the stable regions and the switch regions.

Using DMA has confirmed the importance of the alignment variable in the SPP model. The clustering exhibited from DMA is much more pronounced than from PCA (see **Fig. 3.5** and **3.10**). Therefore, the second eigenvector from DMA organizes the data better than the first principal component from PCA.

4 Setting Locusts to Force Switches

Finally, we can see what is required to force the swarm to change its alignment. The most trivial way to accomplish this would be to set the velocity of each locust to the desired alignment. This represents an unrealistic and inefficient approach. Often, the combined actions of just a few individuals can lead to a switch in direction; therefore, it is useful to find the minimum number of locusts that must be set in order to guarantee a switch will occur.

At each time step of the simulation, the velocity of the locust furthest away from the target alignment (that is, the opposite of the sign of the current alignment) was set to the target alignment (±0.9). The SPP simulation was run from this new starting state for 20 time steps for 50 trials, and the fraction of trials that successfully switched was recorded. Then the next furthest locust was set to the target alignment, and the process was repeated. If more than 90% of the trials indeed ended with a switch in alignment, the number of locusts required was recorded as the critical number for that time step.

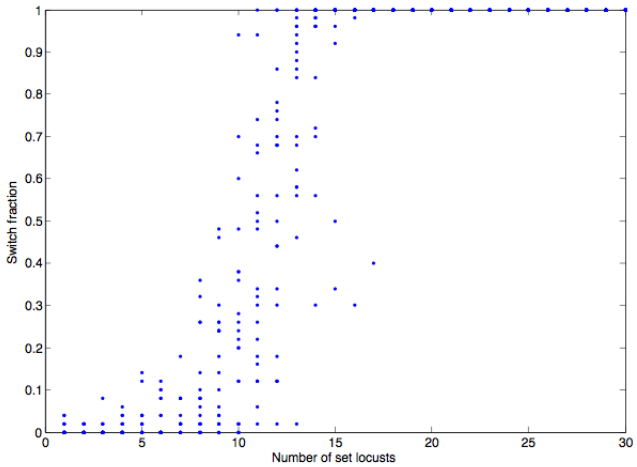


Figure 4.1: The switch fraction behaves as a sigmoid-like curve as the number of set locusts increases. Data are taken from times t = 1600 to 1620 inside a stable region.

In stable regions, the switch fraction has a sigmoid-like ("S" shape) dependence on the number of locusts set to the target velocity. **Fig. 4.1** shows the data from t = 1600 to 1620, a stable region. The switch fraction stays low until about 9 locusts are set, after which it sharply increases, passing 90% when about 14 locusts are set, and then levels off at the maximum value of 1.

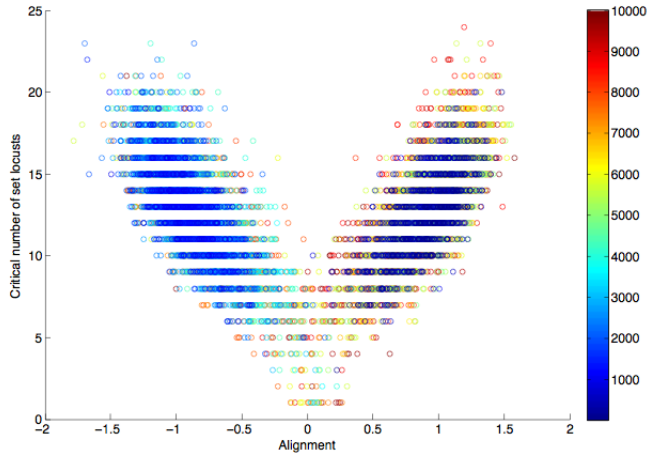


Figure 4.2: The critical number of set locusts required to initiate a switch in alignment increases as the alignment moves from zero. The color bar represents the time of the simulation.

The alignments and the changes in critical number over time for the first 10,000 steps of the simulation are shown on **Fig. 4.2**. The critical number remains high in stable

regions and decreases near switch points or where the alignment fluctuates near zero. When the alignment is near zero, setting approximately 6 locusts, 20% of the population, to the target alignment is enough to trigger a switch with 90% probability (see **Fig. 4.3**). The dependence of the switch fraction on the alignment and number of set locusts is shown further in **Appendix K**. As the alignment approaches the extreme values, however, a higher portion of the population must be set to trigger a switch.

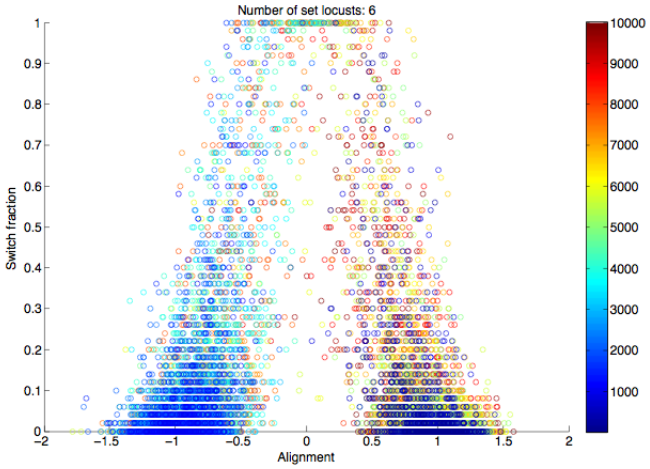


Figure 4.3: When 6 locusts are set to the opposite alignment given that the swarm is already between an alignment of -0.5 and 0.5, the switch fraction increases dramatically. The color bar represents the time of the simulation.

This behavior is expected because when the alignment is near ±0.9, the system is stable and the probability of a spontaneous switch is very low. As the alignment approaches zero, the system is in an unstable state, and will either return to its original alignment or switch to the opposite alignment. With this information, we know we can partially control the outcome of a switch when the alignment is near zero. By waiting until the alignment nears zero and then influencing certain locusts, a switch can be forced to happen with greater predictability.

5 Conclusion

Complex group dynamics are present in a high number of natural phenomena. Fish schooling, bird flocking, and locust swarming are all common occurrences in which choices made by the individual create collective behavior [2,3]. It is often difficult or nearly impossible to understand all the microscopic forces that act to create emergent behavior. Simplified models have been developed that surprisingly create very similar group behavior as seen in nature [5]. As these models are further developed and important variable are discovered, natural systems become better understood.

The SPP model pioneered by Vicsek *et al.* (1995) manages to replicate the swarming behavior seen in locusts through a simple simulation. By probing the model to discover important variables such as locust density and alignment, the behavior of locusts is now more predictable [6]. Szeto's work on the effect of a leader locust was done entirely in a simulation, yet the effects of a leader locust in the real world may be similar [27[.

This study sought to identity any underlying structures in the higher dimensional dataset created by 30 locusts each with a position and a velocity. A new metric measuring how likely a given configuration is likely to switch, the switch fraction, was developed. By comparing several statistical quantities, it was found that a positive relationship exists between the standard deviation and the switch fraction.

Then, dimensional reductions techniques were used to search for lower dimensional substructures within the data. PCA and DMA both identified a variable corresponding to the alignment as contributing the most to the variance of the data. Indeed, the alignment is an effective and informative measure of the state of the swarm.

Finally, the velocities of individual locusts were artificially set to see if a switch could be forced. The number of set locusts required to force a switch at least 90% of the time was called the critical number, and it showed a strong correlation with the alignment variable. The knowledge gained from how to force a switch in this simulation could be applied to the larger scale problem how to disperse swarms of locusts.

Future research in simulations such as the SPP model may provide many practical solutions to problems faced by large portions of the planet. A deeper understanding of what makes swarms change directions or break up could be applied to the swarming of locusts to mitigate damage to crops. Even the collective behavior of crowds of people may be better understood through related research. Forcing a change in the velocity of a locust in a simulation could be mimicked in the real world by having an asymmetric distribution of knowledge of an attractive resource, such as food. Is love a similar resource for the complex emergent behavior observed in humans? Further exploration into the group dynamics generated by individual components could help untangle these complex problems.

Appendix A SPP Model Matlab Code

```
% SppModel.m
% Nick Lavrov '15
% Princeton University
% Code modified from Szeto 2009

function [x, u, alignment] = SppModel(numLocusts, N, dt, r,
length)
% SppModel(numLocusts, N, dt, r, length)
% Runs SPP simulation of numLocusts locusts
% for N steps with time step dt, interaction
% radius r, and length domain length.
% Returns a matrix of all positions x,
% all velocities u, and alignments.
% Typical values: SppModel(30, N, 1, 4, 90)

beta = 1; % parameter in velocity calculaton
xi = 3*sqrt(dt); % noise term

% Vectors representing each locust particle
x = zeros(N, numLocusts); % position of each locust
u = zeros(N, numLocusts); % velocity of each locust

% Initialize the locusts with random positions and
velocities at start
x(1, :) = 10*rand(1, numLocusts);
u(1, :) = randn(1, numLocusts);

% Average velocity at each step is the alignment
alignment = zeros(1, N);
alignment(1) = sum(u(1, :)) / numLocusts;

% Main loop
for i = 2:N
    % Update position
    for j = 1:numLocusts
        x(i, j) = mod(x(i-1, j) + dt*u(i-1, j), length);
    end
    % Update velocity
    for j = 1:numLocusts
        % Calculate local average velocity from neighbors
within
        % interation radius
        velocitySum = 0;
        numIR = 0;
        for k = 1:numLocusts
            if ((abs(x(i-1, j) - x(i-1, k)) < r) || ...
```

```
                    (abs(x(i-1, j) - x(i-1, k)) > (length -
r)))
                velocitySum = velocitySum + u(i-1, k);
                numIR = numIR + 1;
            end
        end
        localAverageVelocity = velocitySum / numIR;
        % Derive G from local average velocity
        G = 0;
        if (localAverageVelocity >= 0)
            G = (localAverageVelocity + beta) / (1 + beta);
        end
        if (localAverageVelocity <= 0)
            G = (localAverageVelocity - beta) / (1 + beta);
        end
        % Find noise term
        Q = (rand()-.5) * xi;
        % Update new velocity
        u(i, j) = u(i-1, j) + dt*(G - u(i-1, j)) + Q;
    end
    % Calculate alignment at this time step
    alignment(i) = sum(u(i, :)) / numLocusts;
end
```

Appendix B Switch Points Matlab Code

```
% switchPoints.m
% Nick Lavrov '15
% Princeton University

function [switchPointsArray, alignmentArray] =
switchPoints(alignment)
% Returns indices of the alignment array and the alignment
sign
% where switches have started and ended.

% Constants
startSwitchThreshold = 0.0;
endSwitchThreshold = -0.5;
stabilityLength = 40;
dropFirstStep = 100;

% State checkers
alignmentState = 1;
isStabilized = false;
stableCounter = 0;

% Initialize arrays to store switch points
switchPointsArray = zeros(numel(alignment),1);
alignmentArray = zeros(numel(alignment),1);
currentSwitchIndex = 1;
tempSwitchStart = 1;
tempSwitchEnd = 1;

% Make sure not to start at a switching point
% Get initial alignment state
while ~isStabilized
    alignmentState = sign(alignment(dropFirstStep));
    a =
alignment(dropFirstStep:dropFirstStep+stabilityLength*2);
    b = sign(a) == alignmentState;
    if (~all(b))
        dropFirstStep = dropFirstStep+stabilityLength;
    else
        isStabilized = true;
    end
end

% Main loop
for i = dropFirstStep:numel(alignment)-stabilityLength
    if sign(alignment(i)) ~= alignmentState
        % start counting to see if stable state is long
```

```
enough
        stableCounter = stableCounter + 1;
        if stableCounter > stabilityLength
            %go back until a<-0.5, end
            %go back until a<0.0, start
            j=i;
            if alignmentState == 1
                while  alignment(j) < endSwitchThreshold
                    j = j - 1;
                end
                tempSwitchEnd = j;
                while  alignment(j) < startSwitchThreshold
                    j = j - 1;
                end
                tempSwitchStart = j;
            end
            if alignmentState == -1
                while  alignment(j) > -endSwitchThreshold
                    j = j - 1;
                end
                tempSwitchEnd = j;
                while  alignment(j) > -startSwitchThreshold
                    j = j - 1;
                end
                tempSwitchStart = j;
            end
            switchPointsArray(currentSwitchIndex) =
tempSwitchStart;
            switchPointsArray(currentSwitchIndex+1) =
tempSwitchEnd;
            alignmentArray(currentSwitchIndex) =
alignmentState;
            alignmentArray(currentSwitchIndex+1) = -
alignmentState;
            alignmentState = -alignmentState;
            stableCounter = 0;
            currentSwitchIndex = currentSwitchIndex + 2;
        end
    end

end

% Remove zeros at the end
i1 = find(switchPointsArray, 1, 'last');
switchPointsArray = switchPointsArray(1:i1);
i2 = find(alignmentArray, 1, 'last');
alignmentArray = alignmentArray(1:i1);
```

Appendix C Autocorrelation

The results of the autocorrelation calculation are an estimate of correlation of a signal with itself at different moments in time. Given a random variable X with time-independent mean μ and variance σ^2, the autocorrelation function R as a function of lag time τ is given by

$$R(\tau) = \frac{\mathrm{E}\left[(X_t - \mu)(X_{t+\tau} - \mu)\right]}{\sigma^2}$$

where E is the expected value operator and X_i is the value of X at time i.

After being normalized by dividing by $R(0)$, the value of $R(\tau)$ lies between -1 and 1 for well-defined functions, with 1 indicating perfect correlation and -1 indicating perfect anti-correlation. A value of 0 indicates no correlation at all.

Appendix D Switch Fraction Matlab Code

```
% switchFrequency.m
% Nick Lavrov '15
% Princeton University

function [x,u,a,sp,spa,studied,switchPercentAtBlock] ...
    = switchFrequency(numLocusts, N, dt, r, length)
% Runs SppModel with the given inputs, then
% runs switchpoints on the output. Restarts
% simulation with given state at each switch
% point and for 40 points before 30 times and
% finds the percent that result in a switch
% in alignment.

[x,u,a]=SPPModel(numLocusts, N, dt, r, length);
[sp,spa]=switchPoints(a);

trials = 30; % number of times at each initialization
totalStepsBack = 40; % furthest to look before switch point
stepIncrement = 1; % how much to step back from switch
point until total

minBlockSize = 450; % minimum stable size the switch must
be preceded by

totalBlocksStudied = 0;
totalBlocks = numel(sp)/2 - 1;
switchPercentAtBlock =
zeros(floor(totalStepsBack/stepIncrement + 1),...
    totalBlocks);
% each row represents how far back we're going
% each column is an individual block
studied=zeros(1,totalBlocks); % if the region is studied,
record the time

% at each startSwitchPoint in sp
for i = 1:totalBlocks-1
    switchTime = sp(2*i + 1);
    % only go if the block > 450 steps
    if switchTime - sp(2*i) > minBlockSize
        totalBlocksStudied = totalBlocksStudied+1;
        studied(totalBlocksStudied)=switchTime;
        % to keep track of progress, uncomment the line
below.
        % display([num2str(switchTime) ': '
num2str(totalBlocksStudied)]);
        % for each stepIncrement back until totalStepsBack
```

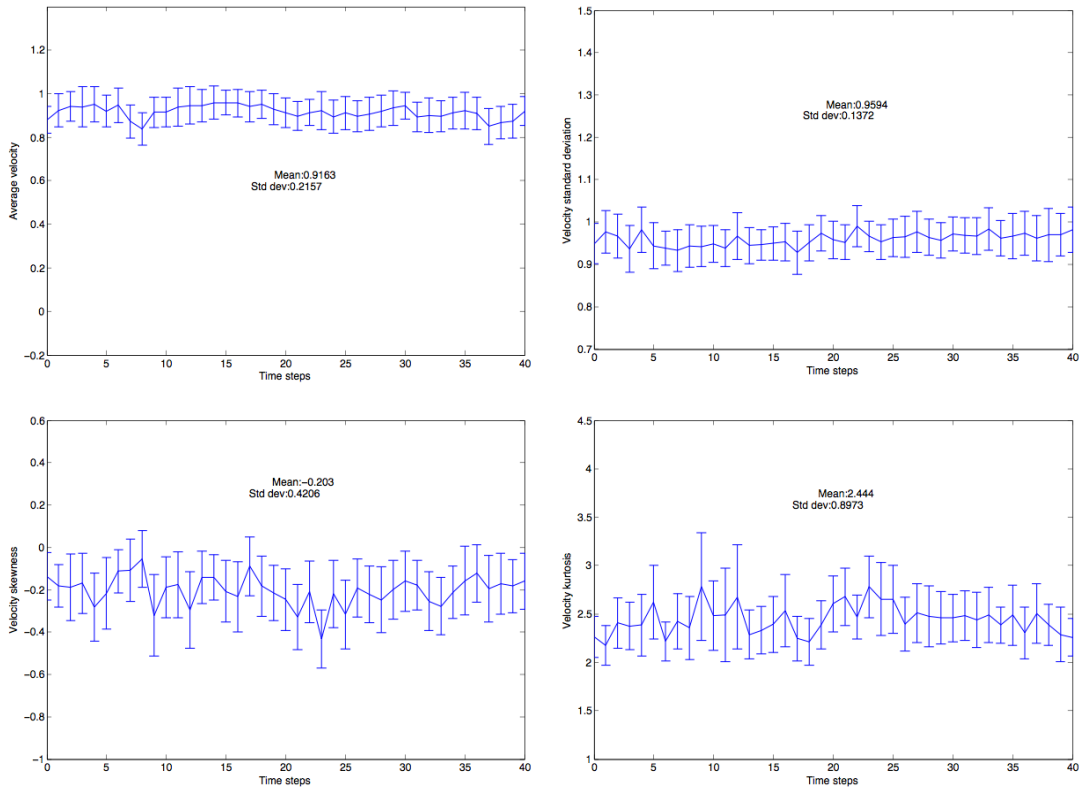
```
        for j = 1:floor(totalStepsBack/stepIncrement + 1)
            timeBack = stepIncrement*(j-1);
            startTime = switchTime - timeBack;
            t = timeBack+25; % run trial up until
switchpoint and then some
            % repeat trials
            for k = 1:trials
                switchPercentAtBlock(j,i) =
switchPercentAtBlock(j,i)+...

switchChecker(x(startTime,:),u(startTime,:),t,dt,r,length);
            end
            switchPercentAtBlock(j,i) =
switchPercentAtBlock(j,i)/trials;
        end
    end
end

% remove zeros from blocks
switchPercentAtBlock =
switchPercentAtBlock(:,any(switchPercentAtBlock));
studied = studied(any(studied,1));
```

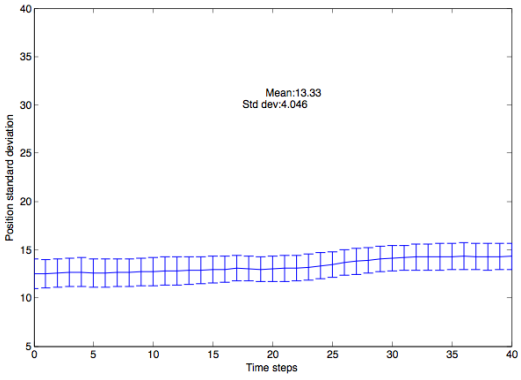
The code for the switchChecker function is below.

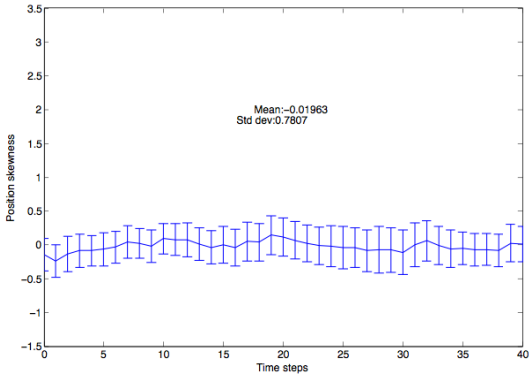
```
% switchChecker.m
% Nick Lavrov '15
% Princeton University


function isSwitched = switchChecker(x, u, N, dt, r, length)
% Runs SPP with given initial state x and u for N steps
% Checks if ending alignment is different from starting
alignment
[~, ~, anew]=initializedSpp(x,u,N,dt,r,length);
isSwitched = sign(mean(u)) ~= sign(anew(end));
```

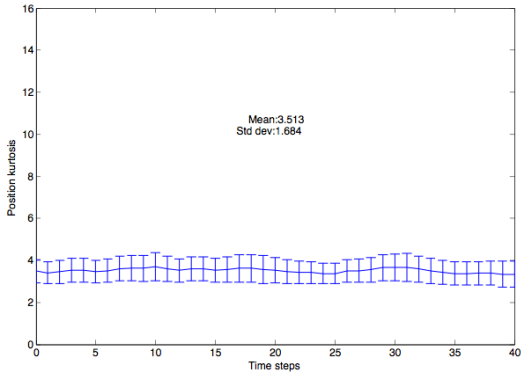
Appendix E Average Values over Stable Regions

The velocity average, standard deviation, skewness, and kurtosis, and the position standard deviation, skewness, and kurtosis are averaged over all 33 stable regions studied and plotted over time. The mean and standard deviation over all the values are summarized in the Table 1 in the Basic Statistical Analysis section. Error bars represent 95% confidence intervals.

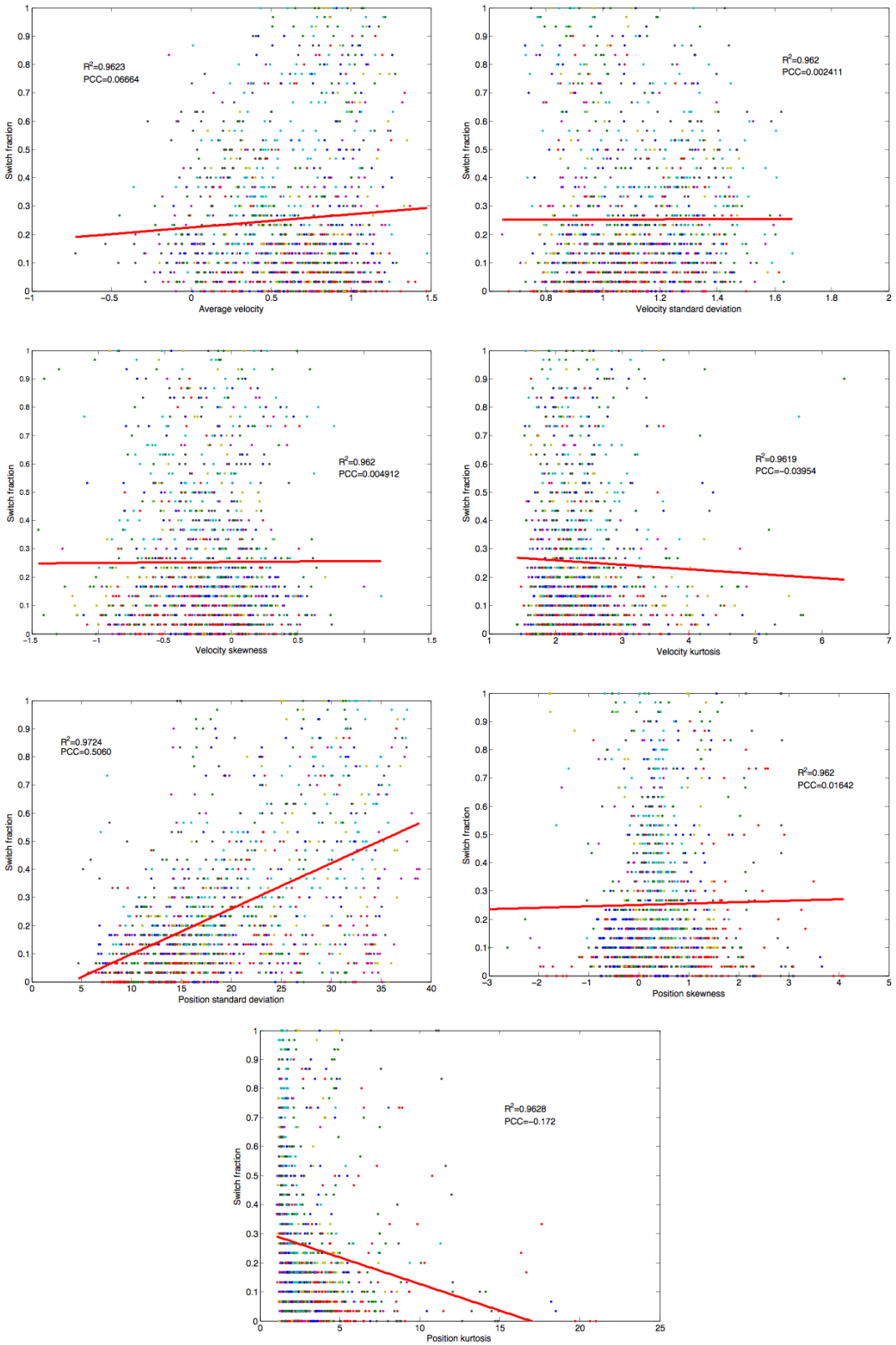



Mean:13.33
Std dev:4.046
Position standard deviation
Time steps


Mean:−0.01963
Std dev:0.7807
Position skewness
Time steps


Mean:3.513
Std dev:1.684
Position kurtosis
Time steps

Appendix F Switch Fraction as a Function of Position and Velocity Statistics



The R^2 and PCC values are summarized in the table below.

	R^2	PCC
Velocity average	0.962	0.0666
Velocity standard deviation	0.962	0.0024
Velocity skewness	0.962	0.0049
Velocity kurtosis	0.962	-0.0495
Position standard deviation	0.972	**0.5060**
Position skewness	0.962	0.0164
Position kurtosis	0.963	**-0.1720**

Appendix G Pearson Product-Moment Correlation Coefficient (PCC)

The PCC measures the linear correlation between two variables *X* and *Y*, giving a value between +1 and -1. A PCC of +1 means *X* and *Y* are totally positively correlated. A PCC of -1 means the variables are totally negatively correlated. A PCC of 0 indicates no linear correlation.

For a sample of size *n*, it is calculated by

$$PCC = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $\bar{x}$ and $\bar{y}$ are the sample means for *X* and *Y* respectively.

A cloud of points would have a PCC of 0. Points arranged in a line with any positive slope would have a PCC of +1. Points in a line with any negative slope would have a PCC of -1. It is important to note that PCC will not represent nonlinear correlations. For example, data plotted in a circle would have a PCC of 0.

Appendix H PCA Matlab Code

```
% Pca.m
% Nick Lavrov '15
% Princeton University

% This code runs assuming that the outputs from
switchFrequency are
% already in the workspace as x, u, a, sp, spa, st, and
perc.
% This is used for the stable regions. With minor
modifications it runs
% for switch regions.

normalizePosConst=22; % used to make position have the same
range as vel
minBlockSize=450;
mem=40; % how long the region to study is

% steps before switchpoint
stimes=-2*mem:-mem;
stimes=stimes*-1;

totalBlocksStudied = 0;
totalBlocks = numel(sp)/2 - 1;
blockTimes = zeros(1,totalBlocks); %holds end time of block

for i=1:totalBlocks
    switchTime=sp(2*i + 1);
    % only go if the block > 450 steps
    if switchTime - sp(2*i) > minBlockSize
        totalBlocksStudied = totalBlocksStudied+1;
        blockTimes(totalBlocksStudied)=switchTime;
    end
end
% remove zeros
blockTimes=blockTimes(blockTimes>0);

% set up raw data
sposFix=zeros(mem+1,numLocusts,totalBlocksStudied);
spos=zeros(mem+1,numLocusts,totalBlocksStudied);
svel=zeros(mem+1,numLocusts,totalBlocksStudied);

% set up SVD matrices
sU=zeros(mem+1,mem+1,totalBlocksStudied);
sS=zeros(mem+1,numLocusts*2,totalBlocksStudied);
sV=zeros(numLocusts*2,numLocusts*2,totalBlocksStudied);
```

```
%% build stableX and stableU data
% go from (endOfBlock-2*mem) to (endOfBlock-mem)

% center all the swarm positions and flip all values to
positive
for i=1:numel(blockTimes)
    switchTime=blockTimes(i);
    if spa(find(sp==switchTime))==1
        % fix x pos so it doesn't wrap and warp the data
        temptime=switchTime-2*mem:switchTime-mem;
        for j=1:numel(temptime)
            tempX=x(temptime(j),:);
            spos(j,:,i)=tempX;
            sposFix(j,:,i)= fixPosition(tempX,length);
        end
        tempU=u(switchTime-2*mem:switchTime-mem,:);
        svel(:,:,i)= tempU;
    else if spa(find(sp==switchTime))==-1
            %because of symmetry, we can convert negalign
to posalign
            temptime=switchTime-2*mem:switchTime-mem;
            for j=1:numel(temptime)
                tempX=-1*x(temptime(j),:);
                tempX=mod(tempX,90);
                spos(j,:,i)=tempX;
                sposFix(j,:,i)= fixPosition(tempX,length);
            end
            tempU=-1*u(switchTime-2*mem:switchTime-mem,:);
            svel(:,:,i)= tempU;
        end
    end
end

% first standardize the vel and position data by
subtracting the mean
% of each column
svelStan=svel-repmat(mean(svel,1),mem+1,1);
sposStan=sposFix-repmat(mean(sposFix,1),mem+1,1);
sposStan=sposStan/normalizePosConst;

% next concatenate the data
sZ=cat(2,sposStan,svelStan);
for i=1:totalBlocksStudied
    [sU(:,:,i),sS(:,:,i),sV(:,:,i)]=svd(sZ(:,:,i));
end

%% PCA plotting
fi=1; % fig number
```

```
for i=1:totalBlocksStudied
    % Embed in 2D space
    figure(fi);
    sT2=sU(:,1:2,i)*sS(1:2,1:2,i);
    scatter(sT2(:,1),sT2(:,2),30,1:41,'o','filled');
    title('PCA embedding');
    xlabel('pc1');ylabel('pc2');c=colorbar('north');
    fi=fi+1;
    % Plot singular values
    figure(fi)
    plot(diag(sS(:,:,i))/sum(diag(sS(:,:,i))),'o')
    title(['Stable Region ' num2str(i) ': Singular value
contribution']);
    xlabel('Singular value number');
    ylabel('Fraction of total');
    axis([0 mem 0 0.2]);
    fi=fi+1;
end
```

The fixPosition function to center the locations of the locusts is below.

```
% fixPosition.m
% Nick Lavrov '15
% Princeton University

function [xnew] = fixPosition(x, length)
% Looks at position at last time step then
% shifts the swarm so that it's not cut off

minCenter=inf;
center=0;

for i=1:length
    temp=sum(min(abs(x(end,:)-i),abs(x(end,:)-i-length)));
    if temp<minCenter
        minCenter=temp;
        center=i;
    end
end
xnew=mod(x-(center-length/2),length);
```

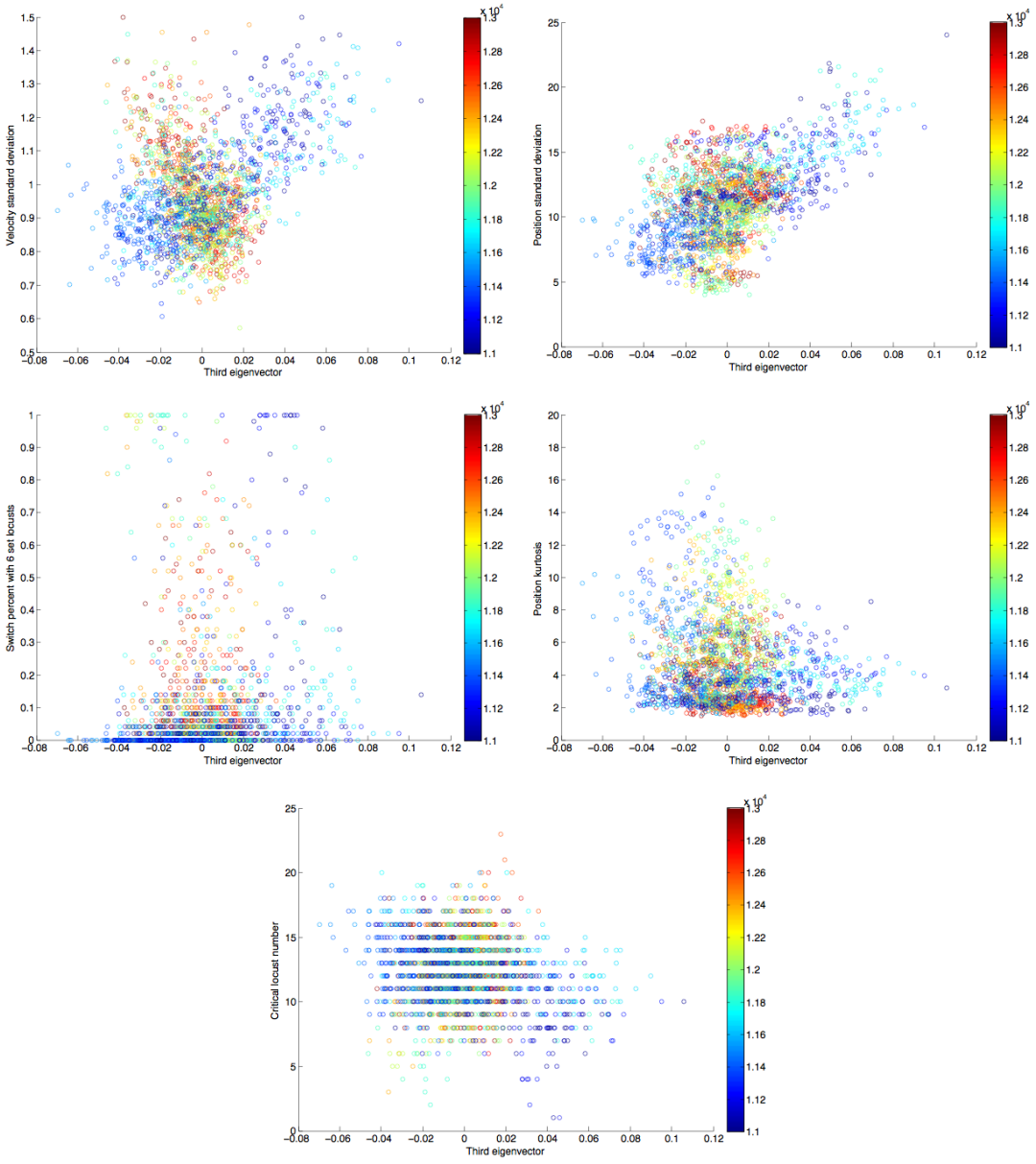
Appendix I DMA Matlab Code

```
% diffusionMap.m
% Nick Lavrov '15
% Princeton University
% Code modified from Wilkes 2013

function [g,v] = diffusionMap(xflip, uflip, epss)
% Returns eigenvalues g and eigenvectors v from DMA
% on a concatenation of x and u data. The gaussian
% kernel uses epss as a parameter
numLocusts=size(xflip,2);
N=size(xflip,1);
length=90;
x = xflip';
u = uflip';

% Converting position data to angle (theta)
theta = 2*pi*(x)/length;
theta_sum = sum(theta);
theta_avg = theta_sum./numLocusts;
theta_avg_rep = ones(numLocusts,1)*theta_avg;
dtheta = minus(theta,theta_avg_rep);

% Normalizing the theta data
sigma = std2(theta);
sigma_rep = ones(numLocusts,N)*sigma;
sigma_inv = 1./sigma_rep;
dtheta_norm = times(dtheta,sigma_inv);

% 3d matrix of dtheta v.s. velocity v.s. time
snapshots = cat(3,u,dtheta_norm);
pos_vel = zeros(2*numLocusts,N);

% Updating with the sorted variables.
for i = 1:N
    snapshot = squeeze(snapshots(:,i,:));
    snapshot2 = sortrows(snapshot,2);
    pos_vel(1:numLocusts,i) = snapshot2(1:end,2);
    pos_vel((numLocusts+1):2*numLocusts,i) =
snapshot2(1:end,1);
end

% Diffusion Map Preparation/Constants
pv_inv = pos_vel';

% Diffusion Map Calculation
dist = squareform(pdist(pv_inv));
```

```
W = exp(-dist.^2/epss^2);

for i=1:N
    A(i,:) = W(i,:)/sum(W(i,:));
end

[v,g]=eigs(A,15);

g=diag(g);[g,j]=sort(g,'descend');v=v(:,j);
```

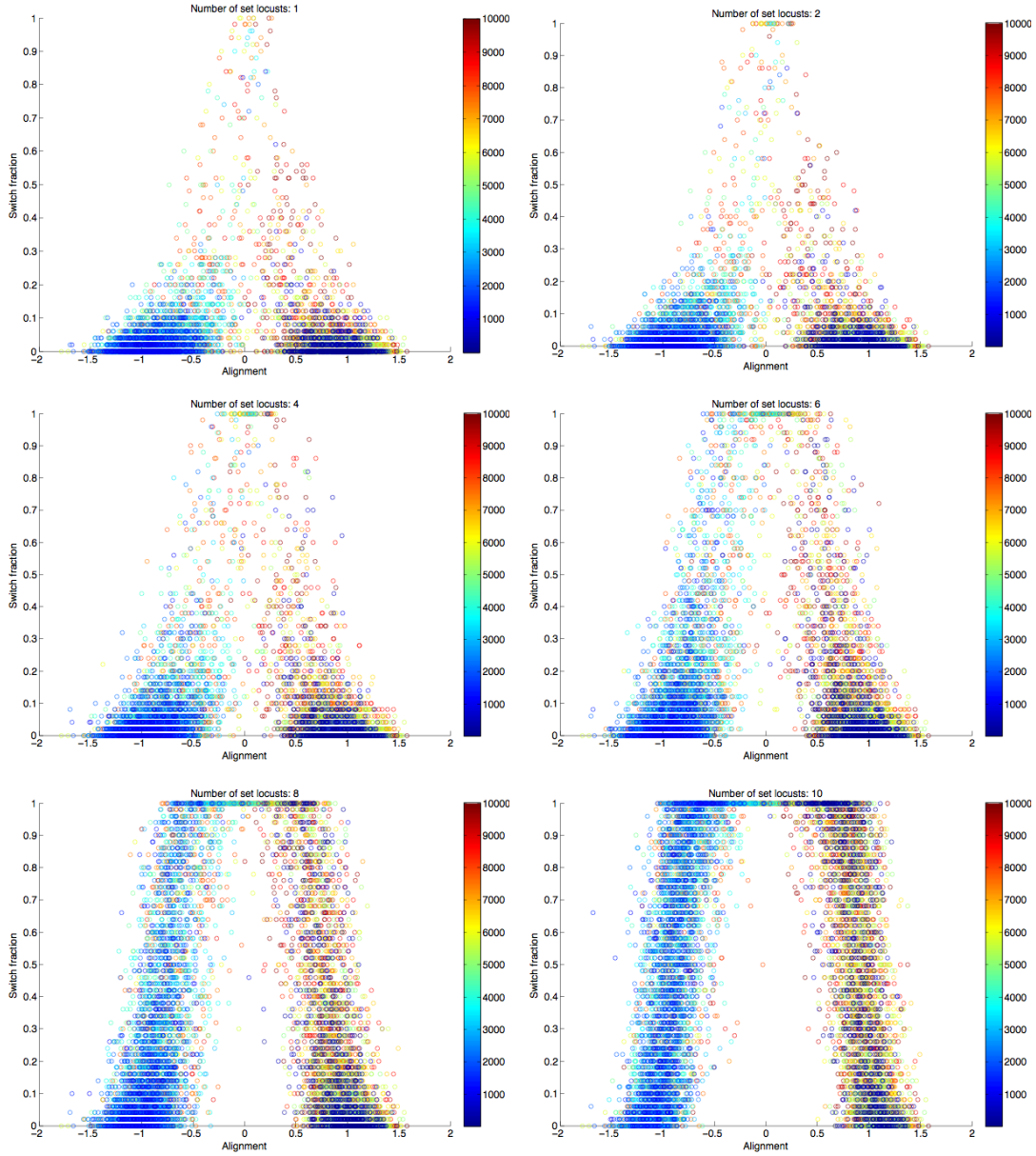
Appendix J Third DMA Eigenvector

If the third eigenvector corresponded with one of these quantities, there would be a linear structure between the two. There is no visible trend. The color bars represent the time in the simulation.



Appendix K Switch Fraction and Alignment for Different Numbers of Set Locusts

As the number of set locusts increases, it becomes more like that a given alignment will switch after 20 steps. The color bar represents the time in the simulation.



Sources

[1] Hameed, M. Locust Swarm 'Like the Plagues of Egypt' Descends Upon Madagascar's Capital. *ABC News*, **Aug. 29, 2014**. http://abcnews.go.com/blogs/headlines/2014/08/locust-swarm-like-the-plagues-of-egypt-descends-upon-madagascars-capital/. Accessed Apr. 24, 2014.

[2] Couzin, I. D., Collective cognition in animal groups. *Trends in Cognitive Sciences*. **2009,** 13(1):36-43.

[3] Godin, J. & Morgan, M. J., Predator avoidance and school size in a cyprinodontid fish, the banded killifish (Fundulus diaphanus Lesueur). *Behavioral Ecology and Sociobiology*. **1985**, 16:105-110.

[4] Cavagna, A. *et al*., Scale-free correlations in starling flocks. *Proceedings of the National Academy of Sciences in the United States of America*. **2010**, 107:11865-70.

[5] Katz, Y., Tunstrøm, K., Ioannou C. C., Huepe, C. & Couzin, I. D., Inferring the struction and dynamics of interactions in schooling fish. *Proceedings of the National Academy of Sciences in the United States of America*, **2011**, 108(46): 18720-25.

[6] Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I. & Shochet, O., Novel Type of Phase Transition in a System of Self-Driven Particles. *Physical Review Letters, The American Physical Society*, **1995**, 75(6):1226-29.

[7] Czirók, A. & Vicsek, T., Collective Motion. *Statistical Mechanics of Biocomplexity*, **1999**, 527:152–64.

[8] Couzin, I. D., Krause, J., James, R., Ruxton, G. D. & Franks, N. R., Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, **2002**, 218(1):1–11.

[9] Cucker, F. & Smale, S. Emergent behavior in flocks. *IEEE. Transactions on Automatic Control*, **2007**, 52(5):852–62.

[10] Uvarov, B. P., *Grasshopper and Locust: a Handbook of General Acridology. Vol. II: Behaviour, Ecology, Biogeography, Population Dynamics*, **1977**, Cambridge University. Press, Cambridge.

[11] Simpson, S. J., Despland, E., Hagele, B. F. & Dodgson, T., Gregarious behavior in desert locusts is evoked by touching their back legs. *Proceedings of the National Academy of Sciences in the United States of America*, **2001**, 98(7):3895-97.

[12] Simpson, S. J., McCaffery, A. R. & Hägele, B. F., A behavioural phase change in the desert locust. *Biological Reviews*, **1999**, 74:461-80.

[13] Rogers, S. M. *et al*., Mechanosensory-induced behavioural gregarization in the desert locust *Schistocerca gregaria*. *Journal of Experimental Biology*, **2003**, 206:3991-4002.

[14] Buhl, J. *et al*., From disorder to order in marching locusts. *Science*, **2006**, 312(5778): 1402-06.

[15] Yates, C., On the Dynamics and Evolution of Self-Propelled Particle Models. Masters Dissertation. **2007**, University of Oxford: Oxford.

[16] Kevrekidis, I. G. *et al*., Equation-Free, Coarse-Grained Multiscale Computation: Enabling Microscopic Simulators to Perform System-Level Analysis. *Communications in Mathematical Sciences*, **2003**, 1(4):715-62.

[17] Erban, R., Kevrekidis, I. G., Adalsteinsson, D. & Elston, T. C., Gene Regulatory Networks: A Coarse-Grained, Equation-Free Approach to Multiscale Computation. *Journal of Chemical Physics*, **2006**, 124(8).

[18] Bingham, E. & Mannila, H., Random Projection in Dimensionality Reduction: Applications to Image and Text Data. *Proceedings of the ACM SIGKDD Internation Conference of Knowledge Discovry and Data Mining,* **2001**, 245-50.

[19] Smith, I., A Tutorial on Principal Components Analysis. University of Montreal. **Feb. 26, 2002**. http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf. Accessed Apr. 20, 2015.

[20] Chatterjee, A. An introduction to the proper orthogonal decomposition. *Current Science*, **2000**, 78(7):808:17.

[21] Holmes, P., Lumley, J. L. & Berkooz, G., *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, **1996**, Cambridge University. Press, Cambridge.

[22] Cusumano. J. P., Sharkady, M. T. & Kimble, B.W., Experimental Measurements of Dimensionality and Spatial Coherence in the Dynamics of a Flexible-Beam Impact Oscillator *Philosophical Transactions of the Royal Society of London*, **1994**, 347:421-38.

[23] Ruotolo, R. & Surace, C., Using SVD to detect damage in structires with different operational conditions. *Journal of Sound and Vibration*, **1999**, 226:425-39.

[24] Liu, P., Equation-Free Analysis for Agent-Based Computation. Ph.D. Dissertation. **2013**, Princeton University: Princeton, NJ.

[25] Coifman, R. R. & Lafon, S., Diffusion maps. *Applied and Computational Harmonic Analysis*, **2006**, 21:5-30.

[26] Nadler, B., Lafon, S., Coifman, R., & Kevrekidis, I. G., Diffusion Maps – a Probabilistic Interpretation for Spectral Embedding and Clustering Algorithms. *Principal Manifolds for Data Visualizatoin and Dimension Reduction*, **2008**, 58: 238-260.

[27] Szeto, B. Computational Models for Locust Swarming Involving Informed Individuals. Bachelors Thesis. **2009**, Princeton University: Princeton, NJ.

[28] Powell, V. & Lehe, L. Principal Component Analysis. *Explained Visually*, **Feb. 12, 2015**. http://setosa.io/ev/principal-component-analysis/. Accessed Apr. 23, 2015.

[29] Tenenbaum, J. B., de Silva, V. & Langford, J. C., A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, **2000**, 290(5500),:2319-23.

[30] de la Porte, J., Herbst, B. M., Hereman, W. & van der Walt, S. J. An Introduction to Diffusion Maps. *Proceedings of the Nineteenth Annual Symposium of the Pattern Recognition Association of South Africa*, **2008**.

[31] Wilks, O. A Comparitive Analysis of Diffusion Maps & the Coarse-Grained Alignment Dynamics of Self Propelled Particle Models. Bachelors Thesis. **2013,** Princeton University: Princeton, NJ.