Simulated Flocking Behavior

Nick Travers
Department of Mathematics, UC Davis
ntravers@math.ucdavis.edu

Abstract:

Flocking refers to the collective and coherent motion of a large group of animals. This project is an attempt to simulate flocking behavior using autonomous agents with simple movement rules. The algorithm is based on the three basic principles first proposed by Craig Reynolds in his ground breaking 1987 paper "Flock, Herds, and Schools : A Distributed Behavior Model" (1). The project consists primarily of visualization modules, although it also contains some preliminary dynamical systems analysis of the flock movement. Unfortunately, due to the high level of complexity of the system standard analysis techniques have been hard to implement. In particular, measuring the degree of instability or chaos in the system via Lyapunov characteristic exponents or an analogus quantity has proved difficult.

Introduction

Although people have always been intrigued by the graceful movement exhibited by flocks of migrating birds, schools of fish, and other such creatures it is only relatively recently that we have begun to systematically study and model this behavior. The first substantial effort to model flocking was probably by the computer scientist and animation artist Craig Reynolds in 1987 (1). However, since that time flocking dynamics has quickly become a subject of great interest among a variety of communities including computer scientists, animal biologists, and applied mathematicians.

Flocking behavior is interesting to scientists for a variety of reasons. Biologists are often interested in studying the flocking behavior of real species in relation to evolution and survival strategies. Herd animals such as Buffalo, for example, run together when attacked by predators rather than dispersing because it increases the average survival rate of animals within the herd. It is a evolutionary strategy. On the other end of the spectrum are the more philosophically inclined AI folks who are interested in studying flocking behavior theoretically as an example of an emergent phenomena and 'collective intelligence'. More recently flocking inspired algorithms have also been used by applied mathematicians and other scientists to solve difficult optimization and data clustering problems (3,4).

My work focuses primarily on simulation methods and visualization of flocking 'boids' on a 2D surface. However, some attempt has also been made to characterize the stability of the system in dynamical systems terms. Preliminary results seem to indicate, quite surprisingly given its complexity, that the system of independent boids is, in fact, a non-chaotic dynamical system, at least within the parameter ranges that generate suitable flocking behavior.

Background

My model was based off the three basic principles of Reynolds which I outline below. The algorithm section contains a more detailed explanation of how these principles were implemented in the 2D model. Reynolds used a 3D model so some adaptations had to be made for the 2D case. No other real knowledge is required outside of basic dynamical systems principles.

(1) Static Collision Avoidance - Boids steer away from stationary obstacles such as walls and other obstructions, and also steer away from other boids that are too close to them.

(2) Flock Centering - Boids try to move towards the center of mass of the flock.

(3) Velocity Matching - Boids try to match their velocity to that of other boids in the flock.

It should be noted that each boid has a limited radius of perception. They can only 'see' other boids that are within this radius. Thus, the algorithm dictates that a given boid actually moves towards the center of mass of its 'neighbors' not the true center of mass of the flock, and only matches its velocity to the average velocity of its 'neighbors' not the actual average velocity of the flock. (Here the neighbors of a given boid refers to all other boids within its radius of perception).

Dynamical System

In dynamical systems terms, a system of N boids can be represented heuristically as a system of 2N symmetric coupled second order differential equations, or 4N first order differential equations. The 'equations of motion' are symmetric because all boids are identical, and second order because the algorithm affecting their movement modifies only their acceleration, not their position or velocity directly. The factor of 2 comes in because the simulation takes places in a 2D world: for each boid there is one equation corresponding to movement in the x direction and one equation corresponding to movement in the y direction.

However, the 'equations of motion' are not really equations. They are given by a reasonably simple algorithm but it requires a number of case statements and would be extremely difficult, if not impossible, to reformulate directly as a system of explicit equations. A schematic outline of the algorithm is given in the following section (it is also well outlined in the code with comments). One unfortunate consequence of the lack of explicit equations, though, is that because there is no Jacobian Matrix we can't evolve orthogonal unit vectors and calculate a set of Lyapunov exponents like we could for a regular set of differential equations.

The Algorithm

At each timestep the following algorithm is applied to all boids simultaneously, and the positions and velocities of all boids at the next time step are updated accordingly. Here neighbors refers to all other boids within our boid's radius of perception, and obstacles refers to all other boids or stationary obstacles within our boid's radius of avoidance. Walls are separate stationary obstacles that confine the boids to a square grid for display purposes. The radiance of avoidance is larger for walls and stationary obstacles than it is for other boids.

The Algorithm: (a = acceleration, v = velocity, p = position)

*v_next = v_current + a_current*timestep*
*p_next = p_current + v_current*timestep + 1/2 * a_current*timestep^2*

where we calculate a_current using the four acceleration factors below:

*a_velocity = (average velocity of neighbors - average velocity of boid) * scaling factor*
*a_center = (average position of neighbors - average position of boid) * scaling factor*
a_static = movement away from the closest obstacle involving both a turning factor and a direct movement away in the direction opposite of the displacement.
a_wall = movement away from the closest section of wall involving both a turning factor and a direct movement away in the direction opposite of the displacement.

*The first three factors are added in order of priority (static collision avoidance > velocity matching > flock centering) until either the maximum acceleration level of a boid MA is reached or until all three factors have been added in. The resulting quantity is our initial determination of a_current. If the boid is within range of a wall this value of a_current is averaged with a_wall to give us a new value of a_current. Finally, if a_current*timestep + v_current exceeds the boid's maximum velocity we cut back the acceleration accordingly so that at the next timestep the boid's velocity will be its maximum potential velocity MV.*

A few notes about the algorithm:

(1) The positions, velocities, and accelerations of the boids are all 2D vectors (x,y).

(2) Because a boid's neighbors change from timestep to timestep the 'equations' change so implementing a Runga-Kutta integrator would be difficult. The factor of 1/2*a*t^2 in the calculation of p_next is an attempt to smooth the integration process, and stay closer to the actual 'equations'. In practice it seems to help.

(3) The turning factors mentioned above are somewhat annoying to calculate (see code) but they appear to smooth the transitions around obstacles and provide a more natural motion. Reynolds actually used a much more complicated turning procedure to replicate 3D flight.

(4) The priority ordering system for the various methods of acceleration was proposed by Reynolds, although he did not specify a precise order in his paper only that they should be ordered rather than averaged. This makes sense if you think about it. Given conflicting impulses a creature must decide which way to move rather than average the impulses and do nothing. For example, a bird can either turn left or right to avoid an upcoming tree, but it would be fatal to average the two impulses and maintain its forward trajectory. Given that we want to prioritize, clearly it makes sense to give obstacle avoidance the highest priority. In practice I found that using velocity matching as the second priority seems to work well.

(5) The scaling for the four different components of acceleration is crucial to the way the system functions. I found it best to keep the magnitude of a_velocity and a_current fixed at 1/3 of the maximum acceleration and compute the magnitude of a_wall and a_static as the inverse of the distance to the obstacle (as you get closer and closer you try harder and harder to avoid it.)

(6) Reynolds actually weighted the effect a boid's neighbors had on it by the inverse of the square of the distance between them. It was not until rereading the paper more carefully that I determined this, and it would have been somewhat difficult to implement in terms of the averaging processes, but the code seems to work at least decently well without this.

Methods

The algorithm outlined above was used to simulate the movements of a set of 20 birds with random initial positions and velocities confined to a square grid. Multiple simulations were run to optimize the model parameters and produce the most realistic flocking movement. Additionally, the stability of the system was assessed by creating very similar sets of initial conditions (boid locations and velocities) and evolving their trajectories to monitor divergence. The simulation engine was run in python and the display module used pygame.

Results

After a good bit of parameter tuning the simulations did finally produce a reasonable approximation of flocking behavior. Randomly distributed boids will cluster into small groups which merge together and eventually form one large flock. Once formed the flock maintains unity as it wanders about the grid for the duration of each simulation. The boids also do a good job of avoiding contact with one another, walls, and other stationary obstacles (Although perhaps they could move around obstacles and away from walls a bit more smoothly). The main failing of the simulations thus far is that the boids don't do a great job of matching their velocity to the average flock velocity. If the velocity data is smoothed over many timesteps the boid's average velocity over the longer time period is close to that of the flock, but at any given instant the velocities may differ substantially. For a demonstration of the flocking process use the program FlockingDisplayObstacles.py with an appropriate data file. For the convergence of flock positions and velocities see Figure 1.

From the simulations you can clearly see that when the flock forms the boids are generally separated by a distance of approximately the obstacle avoidance radius (3 body lengths) and remain in such a relative position structure throughout the simulations. Thus, the system definitely contains at least one attractor - the set of all possible 4N dimensional vectors which have the position components for all boids separated by approximately the radius of obstacle avoidance. Whether this is in fact one attractor or instead has multiple components is not entirely clear. However, because of the way the simulation is run with the flock confined to the relatively small grid which it appears to traverse somewhat randomly, it is likely that all or virtually all of such well spaced position states would eventually be reached - i.e. they must lie on the same attractor. Nevertheless, there may also be other attractors as well with much smaller basins of attraction. Perhaps some highly symmetric set of initial conditions could generate periodic motion of some sort.

As previously mentioned analyzing the stability of the system via standard Lyapunov exponent methods proved difficult because there was no explicit Jacobian matrix. Instead I chose to create flocks with similar sets of initial conditions (only the initial position and velocity of a single boid had been modified) and then evolve the two flocks to monitor divergence in their overall trajectories. The simulations show that even two flocks with such a highly similar initial condition have rapidly diverging trajectories. Within a few thousand timesteps (or roughly the time needed for either flock to traverse the grid a few times) the two flocks have entirely separated. For a demonstration of the divergence processes use the program FlockingDisplayDivergence.py with an appropriate data file.

To be somewhat more quantitative I attempted to calculate a single Lyapunov exponent for the system based upon the magnitude of the displacement, , between the 2N dimensional position vector of one flock and the 2N dimensional position vector of it modified copy as function of time using the formula $= (1/t)*\ln(|_t|/|_o|)$. The velocity components were ignored because, as seen in Figure 1, they appeared to be mostly noise on the single timestep scale at which we were calculating. Unlike the animations which seemed to indicate chaotic behavior (rapidly diverging trajectories of nearby initial conditions) the Lyapunov exponent calculations seemed to indicate that the system was, in fact, non chaotic with a Lyapunov exponent close to zero. Rather than growing exponentially the magnitude of $_t$ appeared to grow roughly linearly in time. Of course, the difference in the overall position vectors was bounded by the grid the flocks were contained

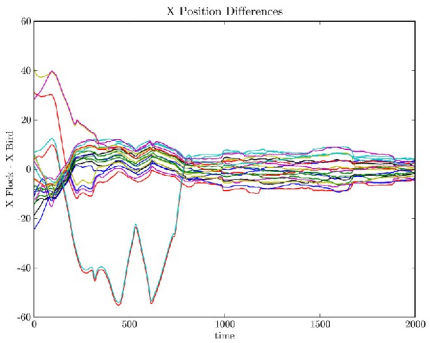
in making them artificially close. However, when the walls were removed the linear growth then became even more pronounced. In retrospect this is actually quite logical. A flock's average velocity should be roughly on the order of the maximum boid velocity MV, and its trajectory more or less straight in the absence of walls or other obstacles, at least over the length of the simulations. Thus, the distance between the centers of two flocks (and thus their corresponding members) should grow roughly linearly in time. See Figures 2,3.

Discussion and Conclusions

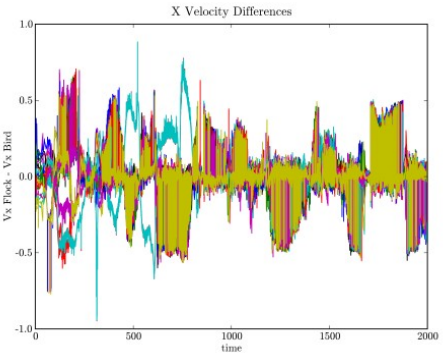
This 2D boid model seems to be a reasonable approximation of flocking behavior, but a number of questions remain to be addressed. First a more accurate quantitative analysis of the degree of chaos in the system is needed. The visualizations give us intuition, but they are not measure able, while the single Lyapunov exponent calculation may be out of place in this context. Secondly, there are number of parameters or ratios of parameters that would be interesting to investigate including the ratio of the radius of perception to the radius of obstacle avoidance, the ratio of a boid's maximum velocity to its maximum acceleration, and the ratio of the number of boids to the size of the grid. The behavior of the system may change dramatically as these parameters are modified.

Figure 1 - Plots of the convergence of boid positions and velocities. (a) The differences in position between individual boids and the center of mass of the flock as a function of time. (b) The differences in velocity between individual boids and the center of mass of the flock as a function of time. (c) The differences in velocity between the individual boids and the center of mass of the flock as a function of time smoothed over 500 timesteps. All plots are for the x-coordinate data only, a similar trend is observed with the y-coordinate due to the symmetry of the system. The positions of the individual boids are seen to converge to the position of the center of mass quickly (i.e. the boids come together in a flock), but the velocity convergence is observed only with the smoothed data.

a.



b.



c.

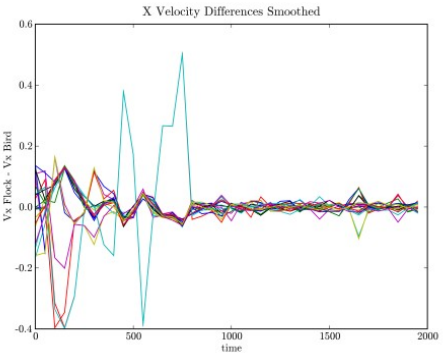
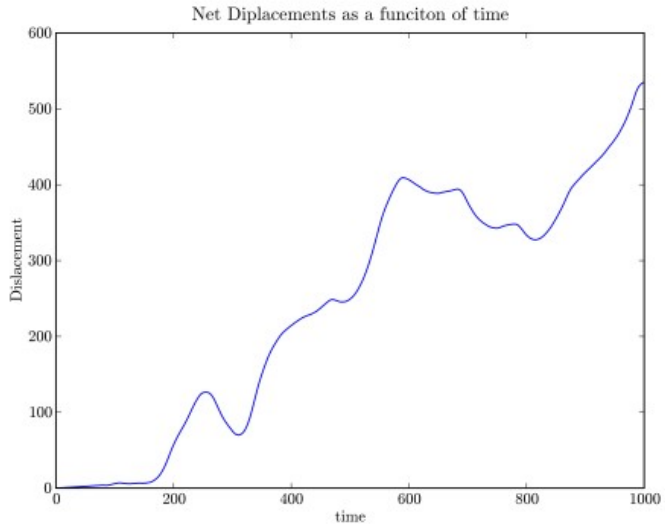


Figure 2 - Divergence Plots, with walls. (a) Plot of the net position displacement between two flocks with similar sets of initial conditions as a function of time. (b) Plot of the Lyapunov characteristic exponent calculated from this displacement as a function of time. In the simulation both flocks were confined to a rectangular grid by imposed walls.

a.



b.

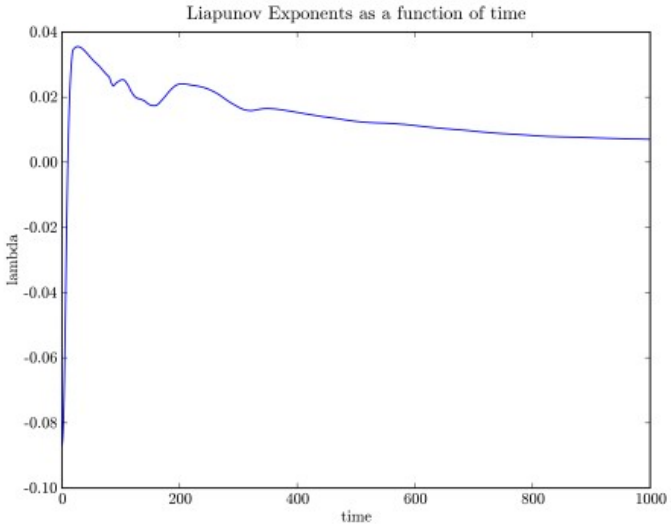
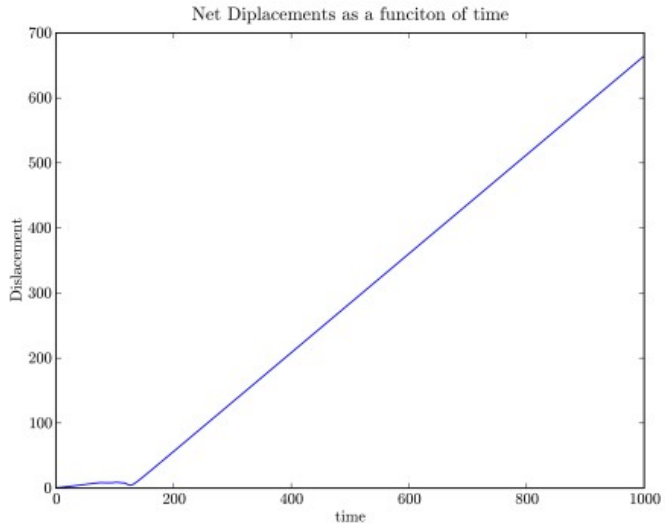
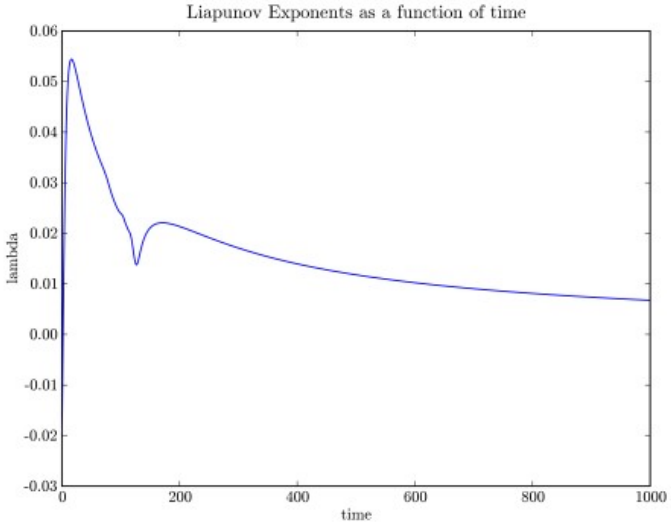


Figure 3 - Divergence Plots, no walls. (a) Plot of the net position displacement between two flocks with similar sets of initial conditions as a function of time. (b) Plot of the Lyapunov characteristic exponent calculated from this displacement as a function of time. The simulation did not involve any walls or boundaries to contain the flock motion.

a.



b.



Bibliography

(1) "Flocks, Herds, and Schools: A Distributed Behavioral Model", Reynolds, C. W. Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34.

(2) Conrad Parker's Boid Page with pseudocode.
http://www.vergenet.net/~conrad/boids/

(3) "A flocking based algorithm for document clustering", Cui et al. Journal of Systems Architecture Volume 52, Issues 8-9, August-September 2006, Pages 505-515

(4) "Particle Swarm Optimization", Kennedy and Eberhart. NeuralNetworks, 1995 Proceedings., IEEE Interantional Conference on.