Angelo Matni
Jonathan Lieberman

Bird Swarm Model

Objective: The goal of this lab is to investigate the dynamics of a simple swarming model governed by the equations:

$$\begin{aligned}\frac{d\mathbf{r}_i}{dt} &= \mathbf{v}_i \\ m\frac{d\mathbf{v}_i}{dt} &= \frac{\bar{\mathbf{v}}_i}{|\bar{\mathbf{v}}_i|} - \mathbf{v}_i - \sum_{j=1}^{N}(\mathbf{r}_i - \mathbf{r}_j)\left(\alpha e^{-\Delta r_{ij}^2/\Delta r_{att}^2} - \beta\frac{e^{-\Delta r_{ij}^2/\Delta r_{rep}^2}}{\Delta r_{ij}^2 + \delta_{rep}^2}\right)\end{aligned}$$

In addition to running the simulation, we investigated problems associated with simulating codes on computers. The problems we investigated are the following: mathematical accuracy of the model, pre-allocating memory to save processing time, checking the validity of the code, error scaling by method, and finally actually investigating the behavior of the model.

Parts 1-2: involved implementing rk4, a function that simulates steps of the differential equation using a 4 part Runge-Kutta estimation. We measured the runtime of this method with different time steps and observed whether pre-allocating the matrices used would alter runtime noticeably.

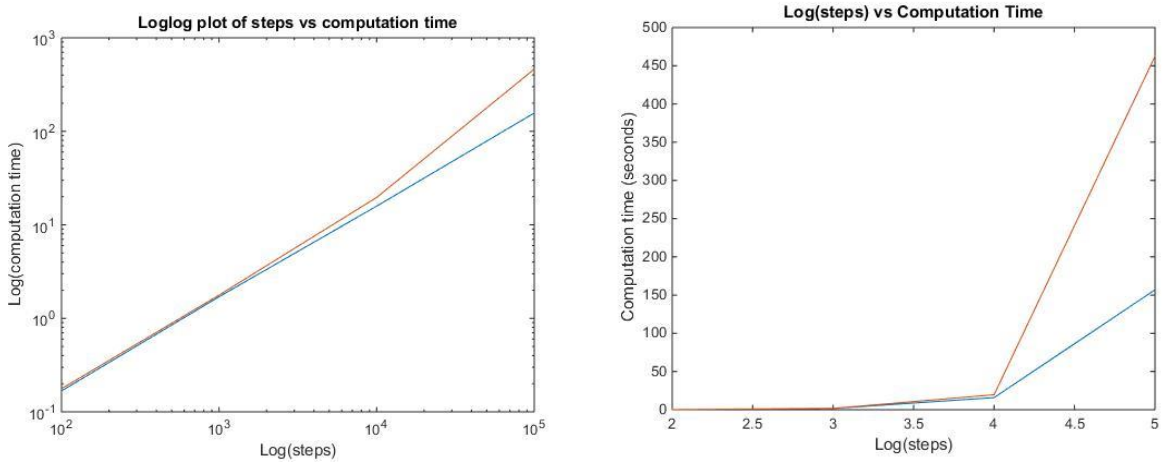


Figure 1a/1b: plots of runtimes for dynamic storage (red line) and for pre-allocation (blue line)

The runtime when matrices are pre-allocated seems to scale linearly with the number of time steps, while the runtime when matrices are not pre-allocated seems to have increasing returns to scale with the number of time steps. This follows the fact that a computer takes increasingly longer to copy a matrix to a new location as the matrix size grows. Pre-allocating becomes much more important as the size of the expected result matrix increases. When using 100000 steps, the computation takes 156.933 seconds when the matrix is pre-allocated and 462.422 seconds when not pre-allocated. Test runs on our more powerful desktop yielded 68.075 seconds with pre-allocation and 240.551 otherwise.

Part 3: To test whether our code runs correctly, we looked at a simple case with 2 boids, where 1 has a fixed position and doesn't move. We also ignored the first two terms in our differential equation for the velocity components. This left us with the force term, which should theoretically lead to one boid orbiting around the fixed one. Due to the similarity to centripetal force equations, the correct choice in initial conditions (as detailed by equations in the question prompt) should yield a perfectly circular orbit.

Angelo Matni
Jonathan Lieberman

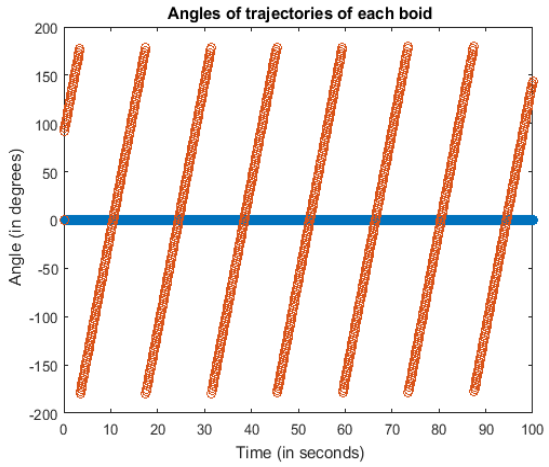


Figure 2: Angles of velocity vectors w.r.t. x-axis

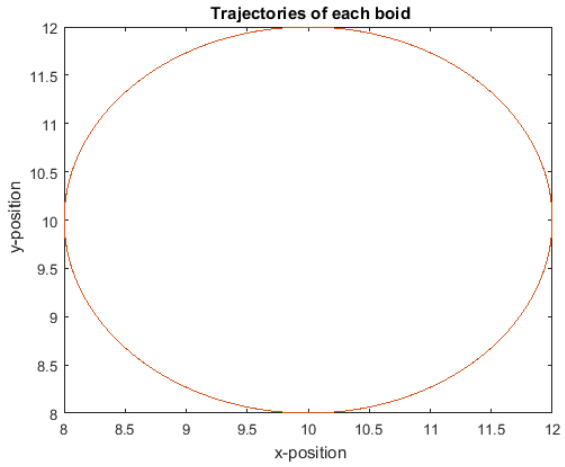
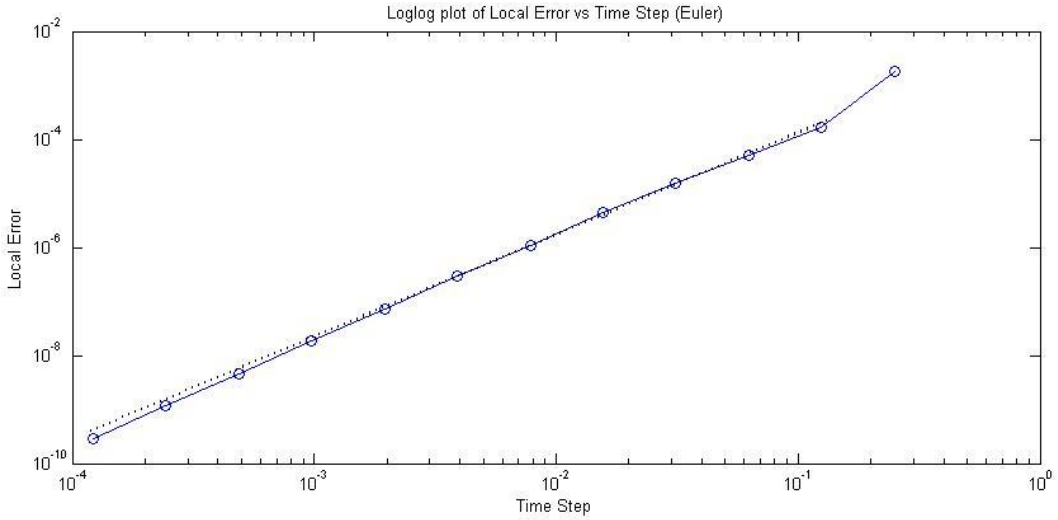


Figure 3: x-y position of the oscillating boid

As you can see, the boid not held fixed oscillated in a circle around the fixed boid, a circle with radius 2. The initial positions of the two boids were (10,10) and (12,10) and the initial velocity term for the red boid was orthogonal to the direction vector from the fixed boid to the moving boid and with just the correct magnitude. While this validation does not guarantee our code is working perfectly, it is a reassuring way to confirm our code against a known solution.

Part 4: An important aspect of differential equation simulations is ensuring our computation approaches the correct value more closely for increasingly smaller time steps taken at a predictable rate. We looked at how our forward Euler and rk4 schemes scaled as the time steps were decreased. Without the exact solution, we could not know the exact error of our results, so we got a general idea for our error by comparing the difference between solutions at each time step with what solutions for 2 steps of half the original time step. It is estimated that the local error for a forward Euler scheme should scale like Δt^2, and the error for an rk4 scheme should scale like Δt^4.



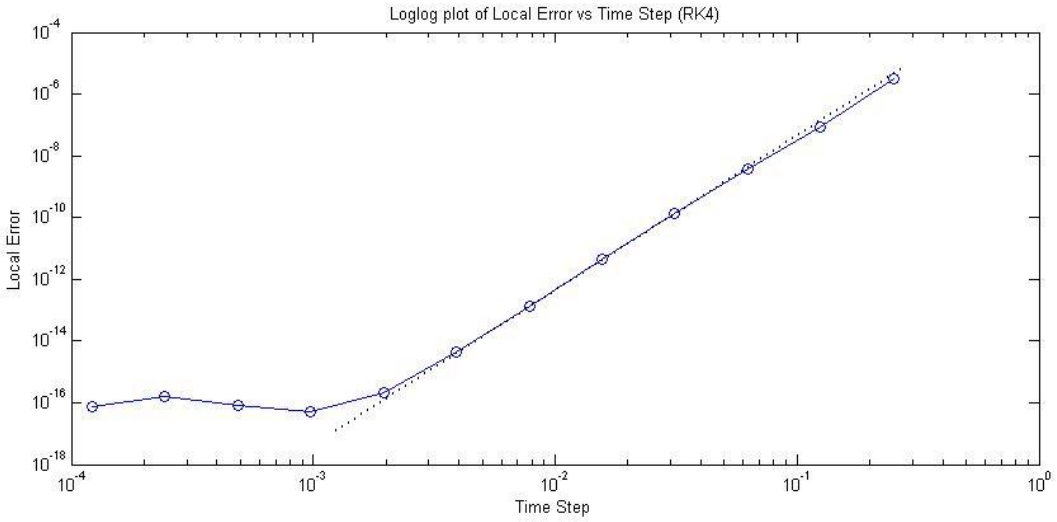


Figure 4: Plots of estimated error between Δt and Δt/2 time steps for the rk4 and Euler methods. Dashed lines show the theoretical error estimates.

The results of our testing seem to indicate that the local error scales similar to theoretical expectations. Also, in the rk4 scheme, the graph seems to flatten out at small time steps. This is likely due to roundoff errors by the computer due to the small size of the numbers. These results seem to show rk4 is better at reducing the local error when the time steps are already relatively small. However, when the time steps are large enough that rk4 and forward Euler local errors are the same order of magnitude, forward Euler is a better scheme. It takes less time to compute and yields comparable errors with rk4. Nonetheless, for errors we are interested in, Runge-Kutta will achieve the desired maximum estimated error in much less computation time.

Part 5: The last thing we evaluated is the actual behavior of the system. After ensuring our code was valid, accurate, and reliably quick, we tested the model, observing how the boids interact.

When alpha and beta are set to zero, the attractive and repulsive forces in the summation do not exist. The only effect the boids feel is from the average velocity of all boids. Eventually, this influence forces all boids to travel in the same direction. This is independent of the boids' starting locations.

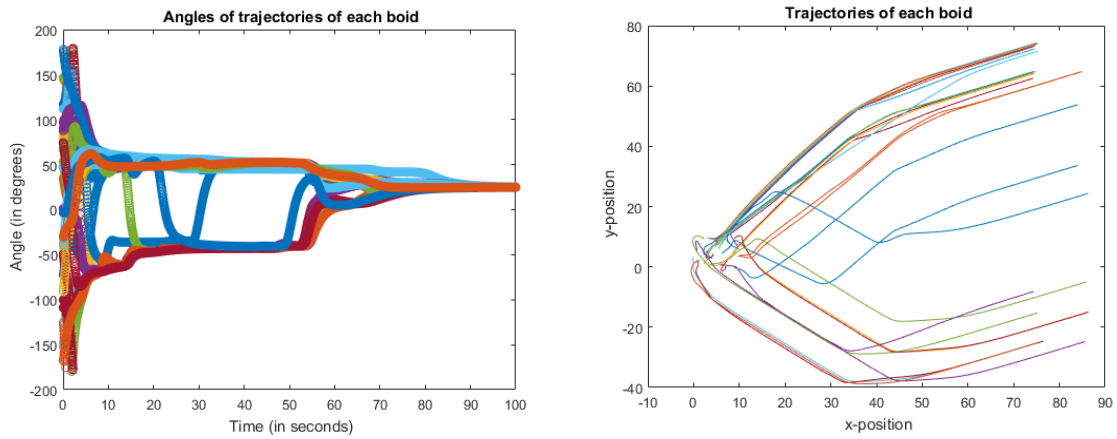


Figure 5: Boid movements for seed 8 random initial positions and velocities.

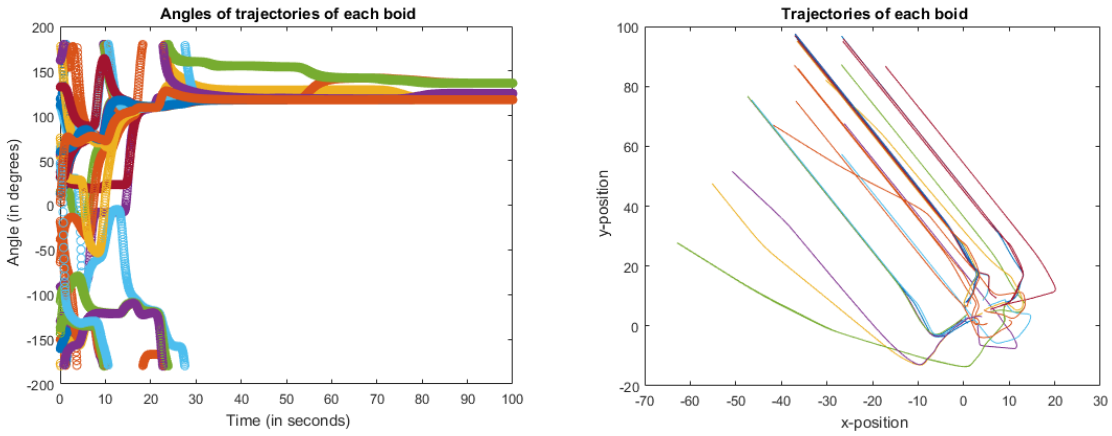


Figure 6: Boid movements for seed 57 random initial positions and velocities.

As you can see, the alignment in direction doesn't happen for a specific angle. Instead, the boids adjust to each other gradually until the boids approach some initial condition specific average direction, or angle of velocity (w.r.t. to the x-axis as before). How soon this happens is also variable. For example, seeds 8 and 23 yielded initial conditions where boids synchronized before the 100 second mark. However, for seeds 17 and 57, this wasn't the case. For this reason, we chose to investigate these initial conditions for longer time intervals and as we decreased the radius of influence of the boids.

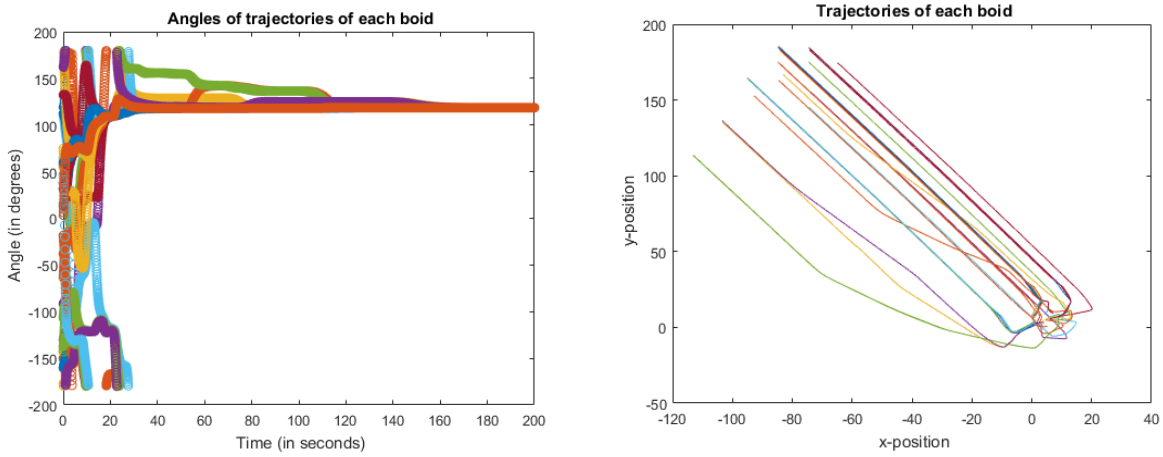


Figure 7: Boid movements for seed 57's initial conditions over 200 seconds instead of just 100 seconds.

The far more diverse initial conditions of this seed led to slower synchronization, but it eventually happened. The first two terms in the velocity differential equation ensure this behavior. Even if two larger boid groups emerge, one will eventually "overpower" the other, dragging those boids' velocities closer in direction to the larger and more tightly packed group.

Angelo Matni
Jonathan Lieberman

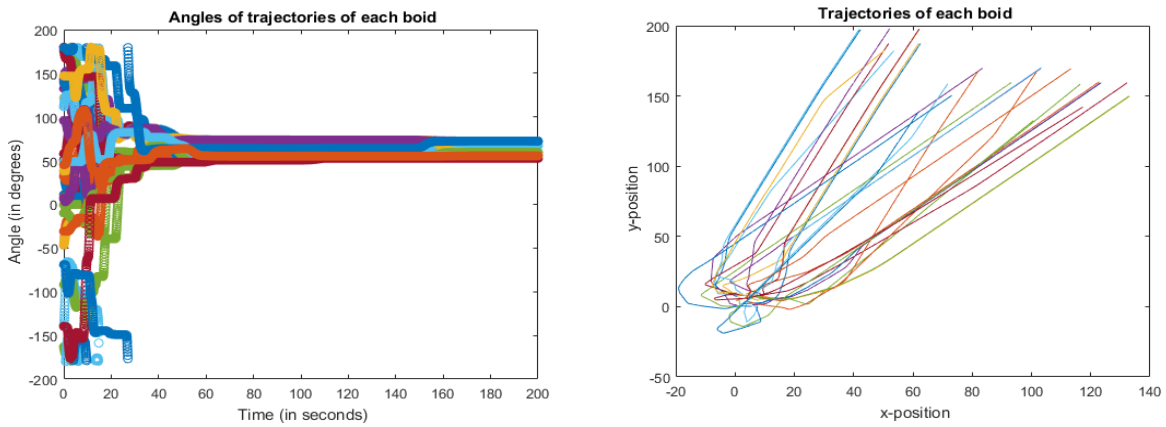


Figure 8: Boid movements for seed 57 ICs and a radius parameter of .5 instead of

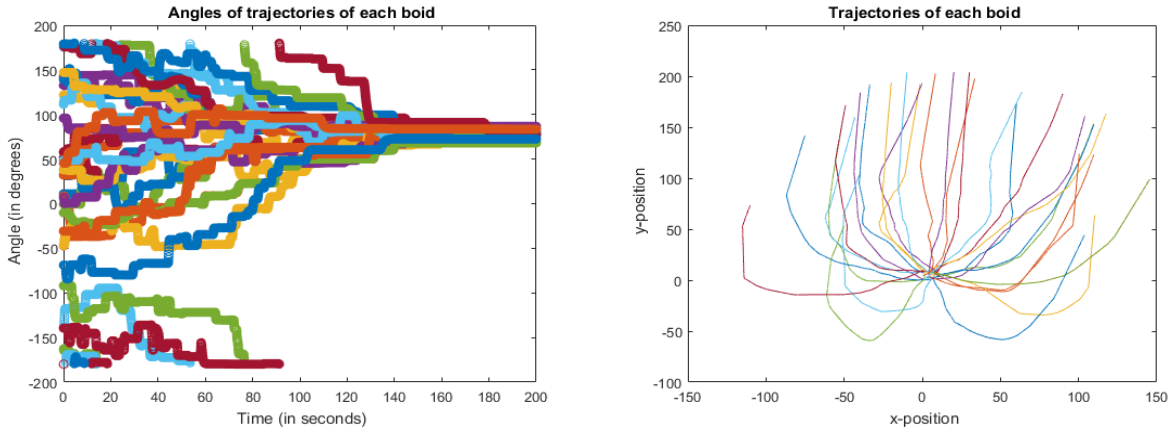


Figure 9: Boid movements for seed 57 ICs and a radius parameter of .25 instead of .5

As the radius parameter decreased, it took longer for boids to be influenced by each other. This is because the boids had to be closer to each other to impact each other's trajectories as easily. A decrease in .5 meant boids also had to be twice as close to maintain equivalent influence over each other. This also means that if two larger groups form, as they did in figure 8, it would take far longer for one group to "overpower" the other. However, end behavior as in figure 7 should still occur.

Now consider boids aren't perfect at synchronizing with each other. Noise can approximate this, and so it was used. The velocity of each boid at each time step was still normalized, but its trajectory was slightly changed by a factor of $\sqrt{\Delta t}$ with a noise parameter. The order parameter defined below

$$V(t) = |\frac{1}{N}\sum_{i=1}^{N} \mathbf{v}_i(t)|$$

measured how synchronized the boids were. A plot of this at the end of the time interval for a run for different noise levels yielded:

Angelo Matni
Jonathan Lieberman

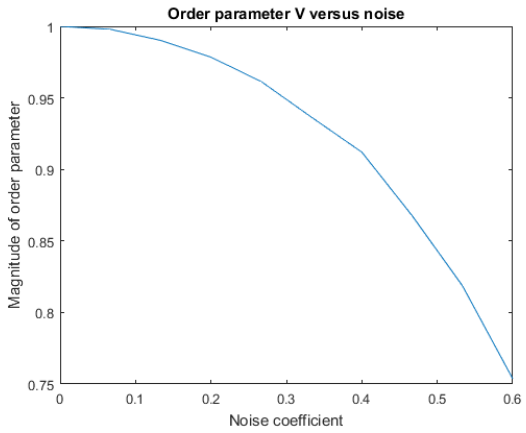


Figure 10: Order parameter versus noise coefficient. 1 corresponds to synchronized, 0 to perfectly de-synced.

When no noise is present, the boids share the same trajectory by the end of a run (for radius coefficient 1 and other parameters as usual), so the order parameter is 1. When noise is introduced, the boids aren't ever fully able to synchronize, so the order parameter decreases. This transition between coherent and incoherent motion grows more rapidly as the noise coefficient increases linearly.

Finally, subjecting the system to all three terms in the velocity differential equation yields more interesting behavior. Grouping and synchronization happens globally, but local boid movements yield vortices. Too high the radius of influence coefficient and vortices will be avoided since the whole boid group will influence every boid member close to equally. Too small a radius coefficient and no group vortices will be stable enough to unfold. A radius coefficient between .3 and .5 produced vortices for entire boid groups most consistently.

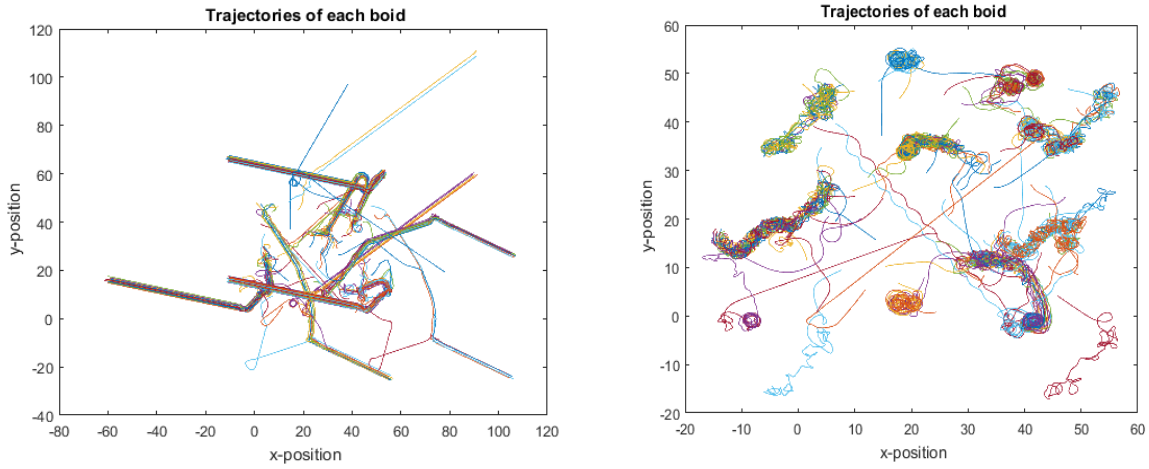


Figure 11: Boid trajectories for radius coefficient of 1 (left) and of 0.1 (right)

A radius of influence coefficient of 1 prevented vortices to form, as entire groups are shown with all their boid members heading in the same trajectory. A radius of influence coefficient of .1 yielded extremely chaotic oscillations for very small boid groups. Neither cases resemble a school of fish type behavior, even if the second produces weaving boid trajectories

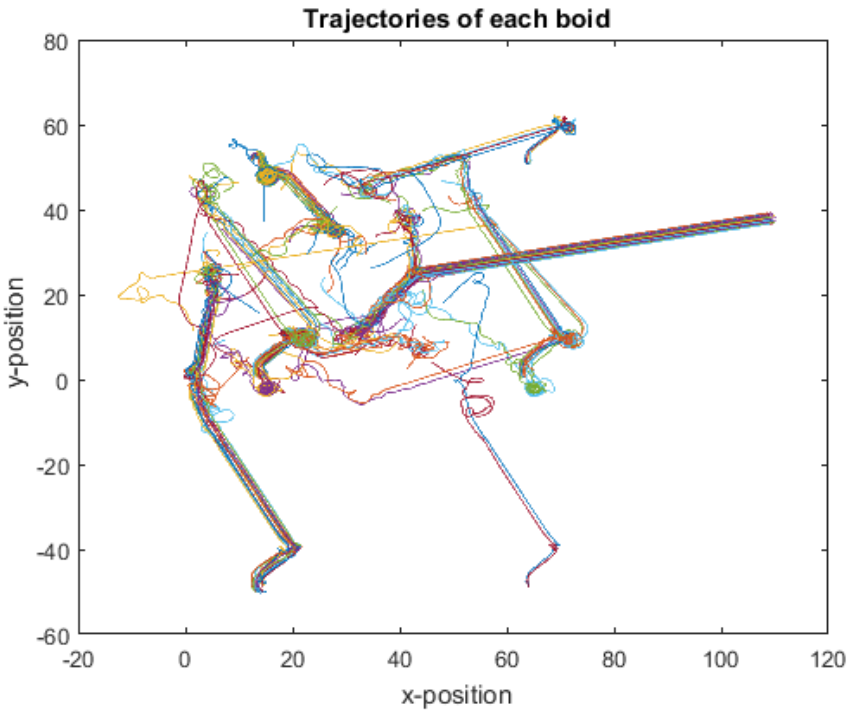


Figure 12: Boid trajectories for radius coefficient of .5

Within the correct range for radius of influence coefficients, vortices form, as seen at positions (20,10) and (20,50), where two separate boid groups began oscillating about their center of mass. This behavior is more akin to a school of fish.

Conclusion: Implementing the rk4 scheme and observing behavior for this swarming model showed us the benefits of a step-by-step analysis with multiple forms of validation. By showing a more trivial consequence of the model in Part 3, we were assured our more complicated term in the differential equation was working as expected. By showing that our rk4 and Euler schemes scaled appropriately for decreasing time step, we were confident in the accuracy of our solutions when analyzing the full model.

We then produced system behavior akin to school of fish movements. By varying the radius of influence parameter, we found two values where: the behavior differs from that of a school of fish in predictable ways, but the range between those values produced the desired effect of vortices. In this way, we approached the model from multiple angles to gain a deeper understanding of the different forms of interactions possible between the boids.