COSC 420 – Biologically Inspired Computation

Project 5

Particle Swarm Optimization

Kelsey Kelley

20 April 2019

Introduction

Why is the sky blue? Why does peanut butter taste better straight out of the jar? Were Ross and Rachel from *Friends* really on a break? Why do birds fly in flocks and fish swim in schools? Most of these questions are hard to understand or even answer. While the answer to the latter may not be fully understood, through the utilization of Particle Swarm Optimization (PSO) interesting theories and inferences can be made. PSO is an optimization algorithm that came about from observing the behaviors of flocking birds and other organisms that are self-organizing. While the original intent of this algorithm was to model the behavior of a simple social system, this algorithm has been modified and even simplified to solve many different problems. This project is based on a PSO algorithm that's goal is to find the maximum of a problem, using the quality function given.

Theory/Methods

The idea of a PSO is to utilize a search space to find the optimum solution to a given problem based on using a number of particles with random initial positions and evaluating these particles based on the quality of their location. This algorithm needed a number of arguments to be passed to it. These values were; number of epochs, number of particles, inertia, cognition parameter and social parameter. To implement this algorithm first given a number of particles their initial positions had to be randomly set, in order to allow a uniform distribution and not introduce bias to the problem. Once the initial position was set, the quality of each particles position was evaluated in order to find the global best position of the swarm. Several distance metrics were used to appropriately evaluate the positions quality.

$$mdist = \frac{\sqrt{max_x^2 + max_y^2}}{2}$$

Figure 1. Calculation for mdist

Where in figure one the x and y subcomponents of max is equal to the world width and world height values.

$$pdist = \sqrt{(p_x - 20)^2 + (p_y - 7)^2}$$

Figure 2. Calculation for pdist

$$ndist = \sqrt{(p_x + 20)^2 + (p_y + 7)^2}$$

Figure 3. Calculation for ndist

In figures two and three the p_x and p_y values indicate the x and y components of the current position of the particle. Using figures one through three the quality of each particle could be calculated. In this implementation of the PSO algorithm there were two functions to be optimized, so therefore there was two different quality functions.

$$Q(p_x, p_y) = 100 * \left(1 - \frac{pdist}{mdist}\right)$$

Figure 4. Quality calculation for problem 1

$$Q(p_x, p_y) = 9 * \max(0,10 - pdist^2) + 10 * \left(1 - \frac{pdist}{mdist}\right) + 70 * \left(1 - \frac{ndist}{mdist}\right)$$

Figure 5. Quality calculation for problem 2

Once the quality of the positions had been calculated, the global best of the population could be found to be used in other calculations. Once the global best was initially found, the main loop of the algorithm could start to execute. The main loop continues until the error values for the x and y points were both less than or equal to 0.01, or the iterations reached the number of epochs. In this continuing loop first the velocity gets updated.

$$v' = inertia * v + c_1 * r_1 * (personal_{best} - position) + c_2 * r_2 * (global_{best} - position)$$

Figure 6. Velocity update function

Where c_1 is the cognition parameter, c_2 is the social parameter, and r_1 and r_2 are two random numbers between [0,1]. Once the updated velocity was calculated, it had to be checked that the new velocity did not exceed the max velocity that was set. If it did the velocity was changed using this function

$$v = \left(\frac{max_v}{\sqrt{v_x^2 + v_y^2}}\right) * v$$

Figure 7. Scaling velocity function

After the velocities for each particle were calculated, the positions were updated.

$$position' = position + v'$$

Figure 8. position update function

Once the positions were updated the quality of all the particles had to be updated and processed again. The global best is updated as well as the personal best for each particle if the quality function deemed it better. After the main loop finished executing, the average error for the x and y coordinates could be calculated.

$$error_x \mathrel{+}= \left(position_{x[k]} - global_{best_x}\right)^2$$

$$error_x = \left(\sqrt{\frac{1}{2 * num_{particles}}}\right) * error\_x$$

Figure 9. Average x error calculation

$$error_y \mathrel{+}= \left(position_{x[y]} - global_{best_y}\right)^2$$

$$error_y = \left(\sqrt{\frac{1}{2 * num_{particles}}}\right) * error\_y$$

Figure 10. Average y error calculation

Through the PSO algorithm, there should be a convergence towards the local maximum within the problems provided. This should take place over the course of a number of iterations along with a steadily decreasing average error. In this project two quality functions were provided. Q1 has one local maximum, and Q2 has two maximum

Data/Results

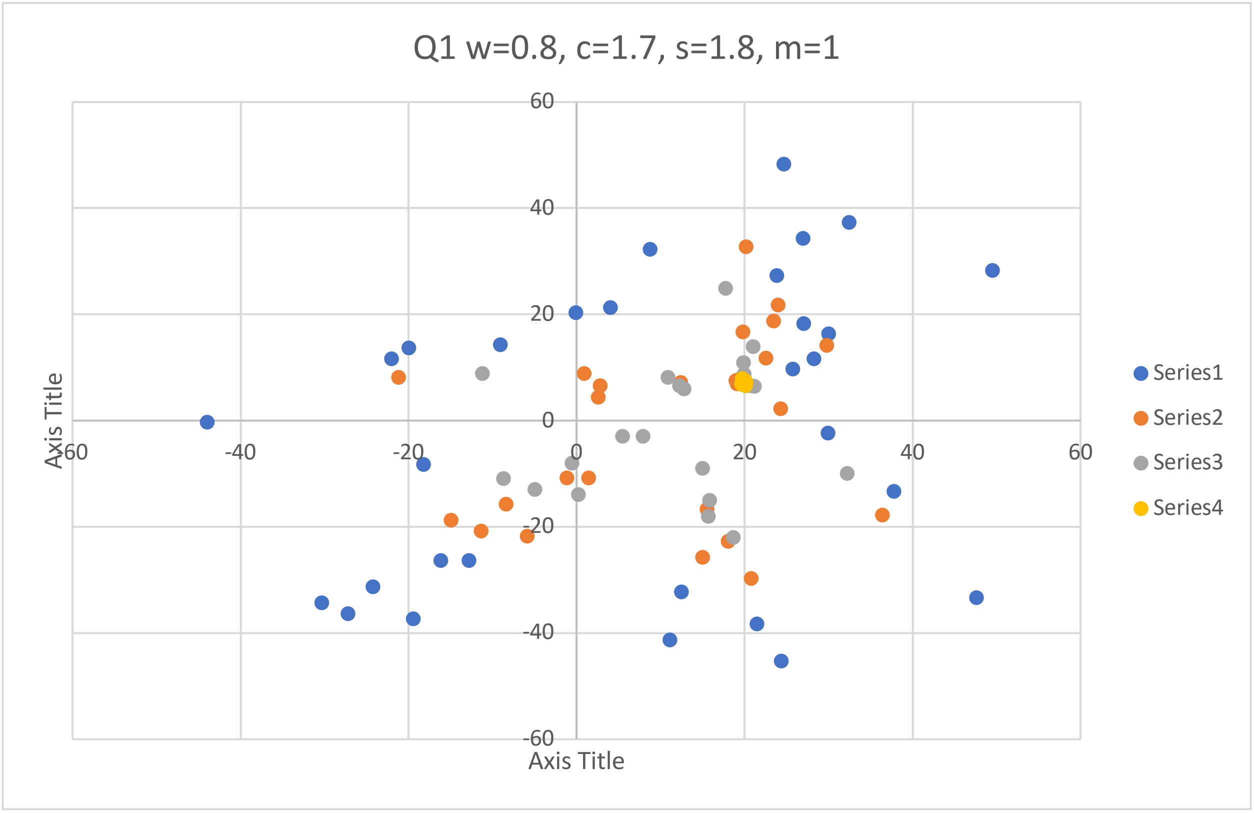

Figure 11. Graph of particles with Q1 function

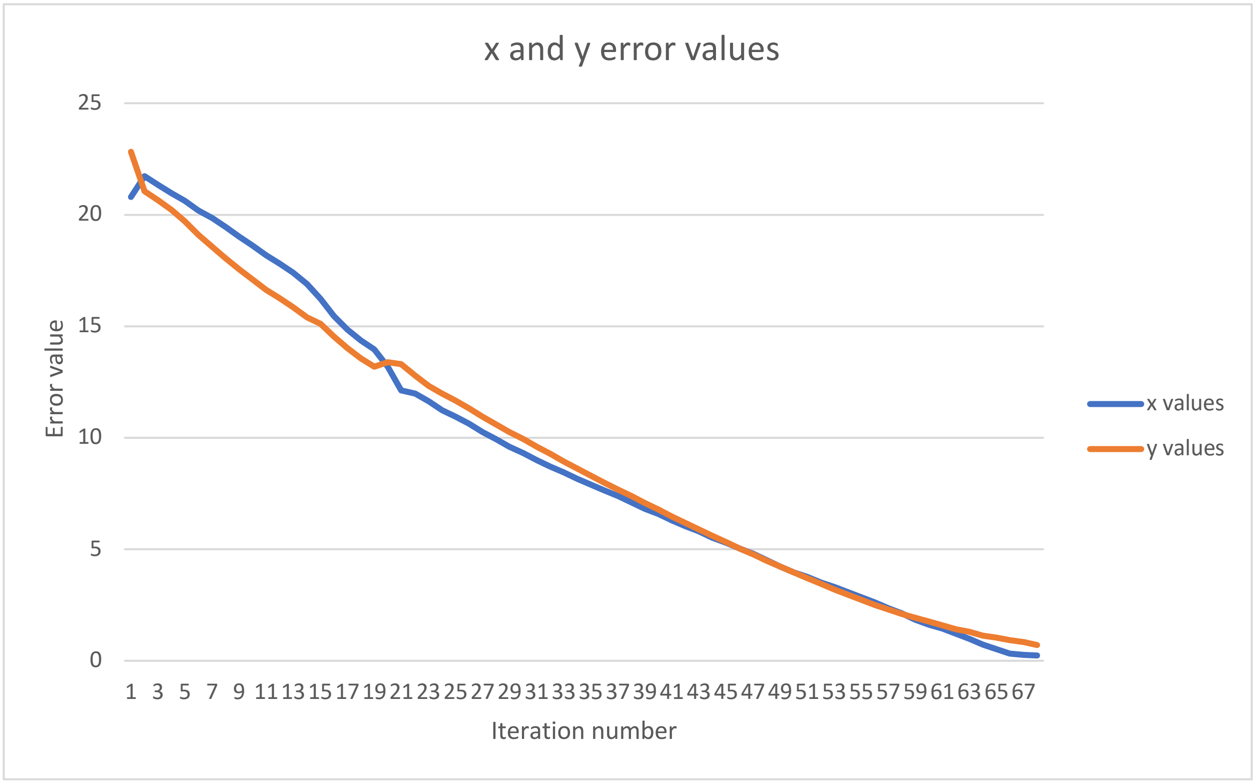

Figure 12. Progression of error values through iterations

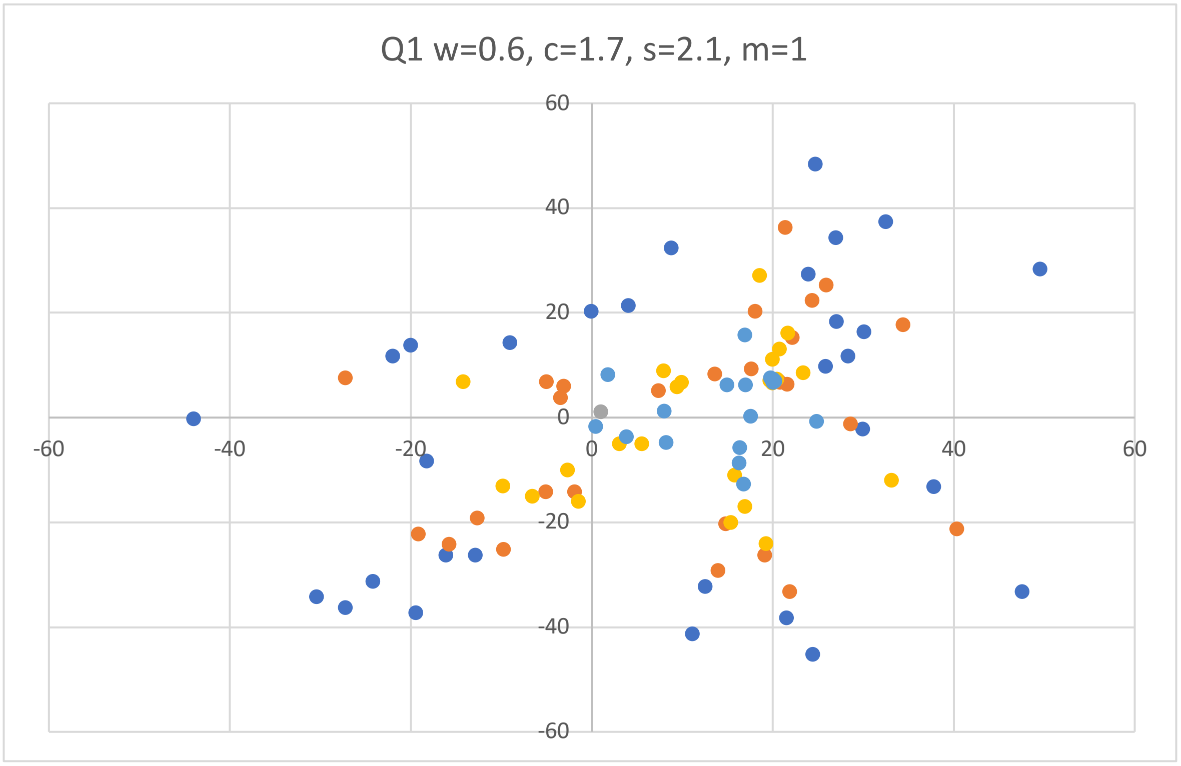

Figure 13. Particles when social parameter was increased

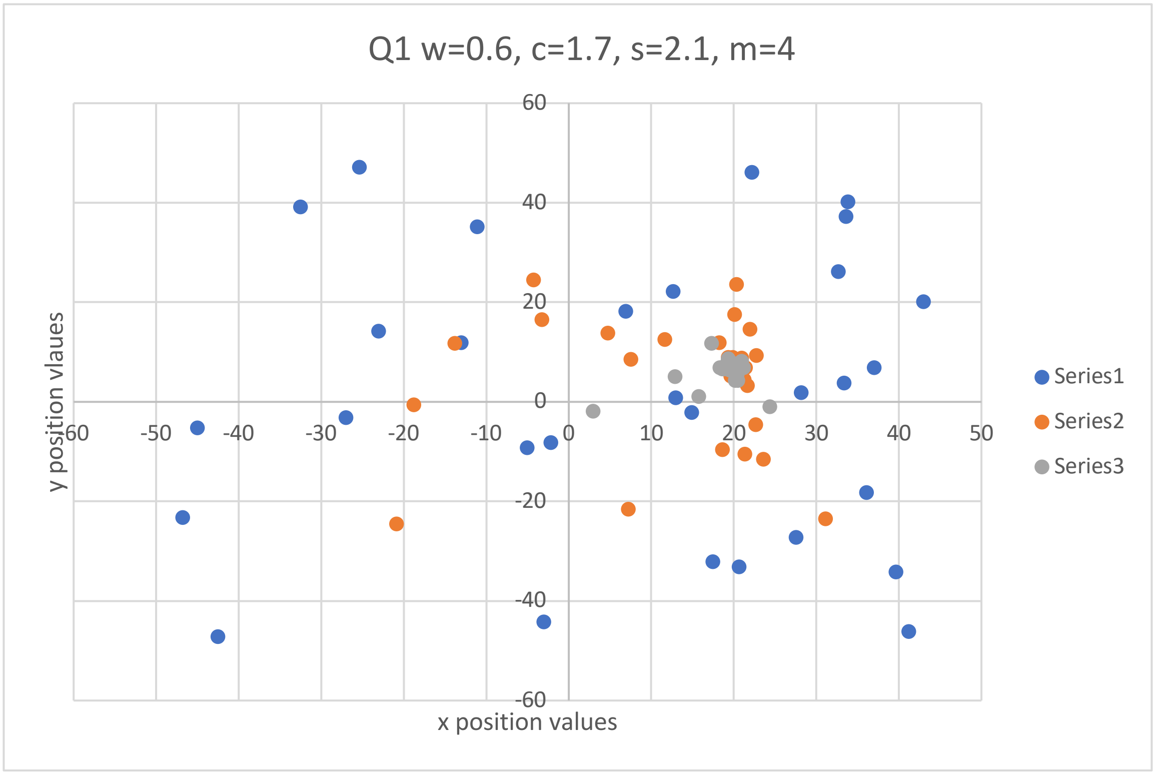

Figure 14. Particles when max velocity was increased

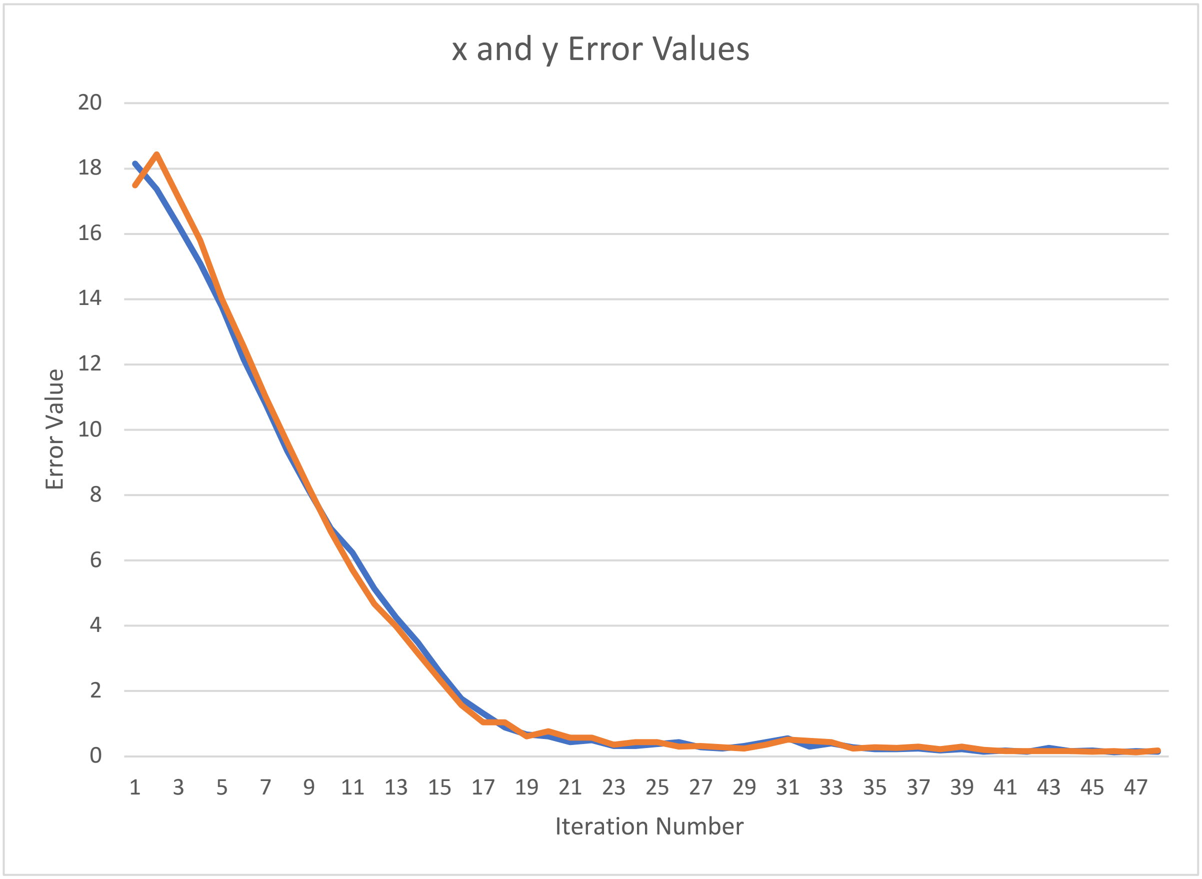

Figure 15. Progression of error values when max velocity was increased

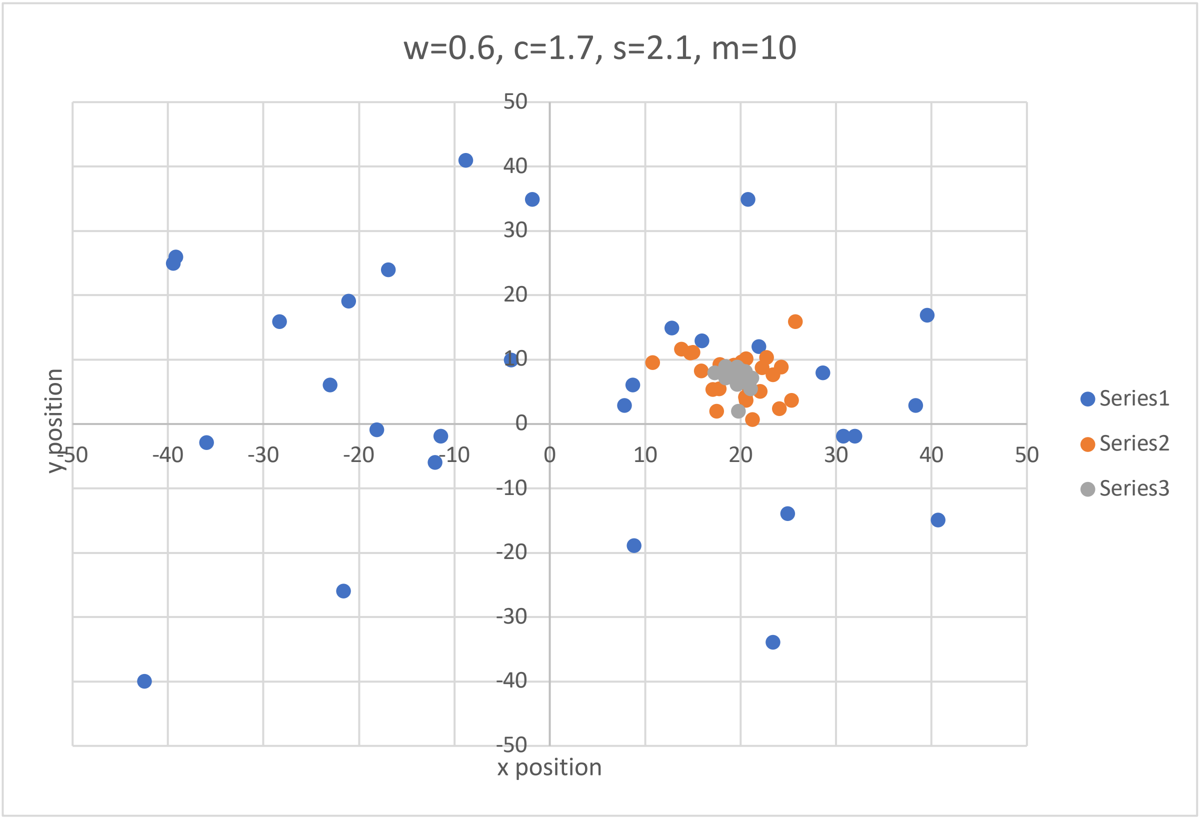

Figure 16. Particles when max velocity was largely increased

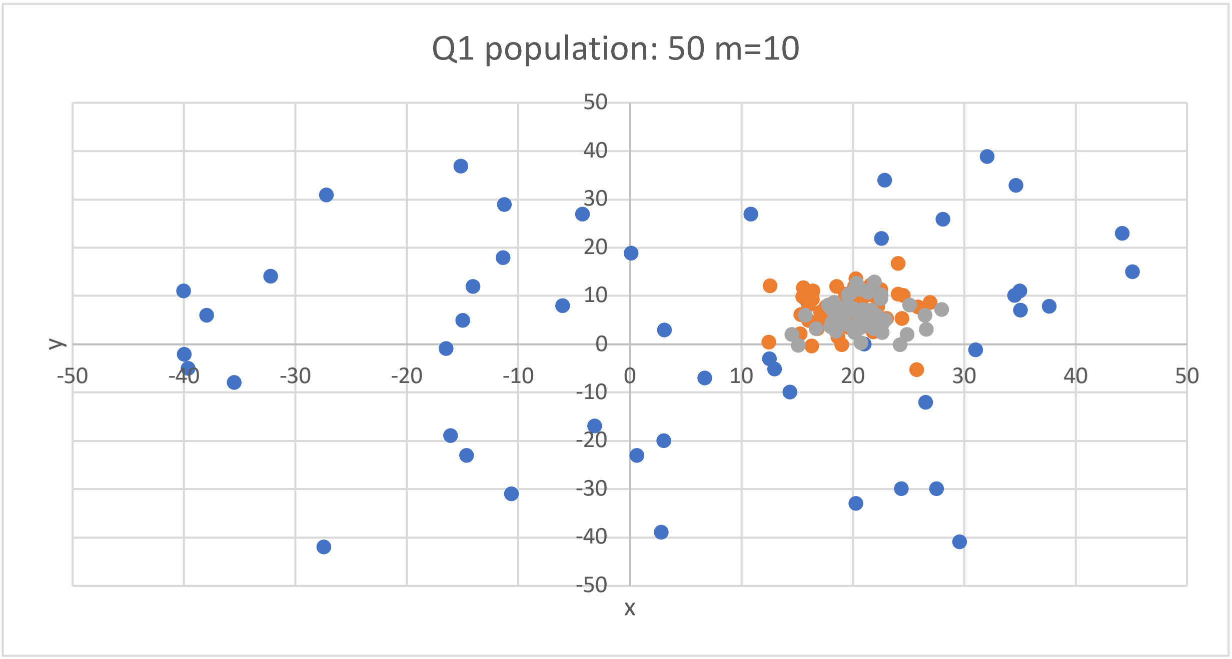

Figure 17. Particles when max velocity was largely increased, and population size was increased

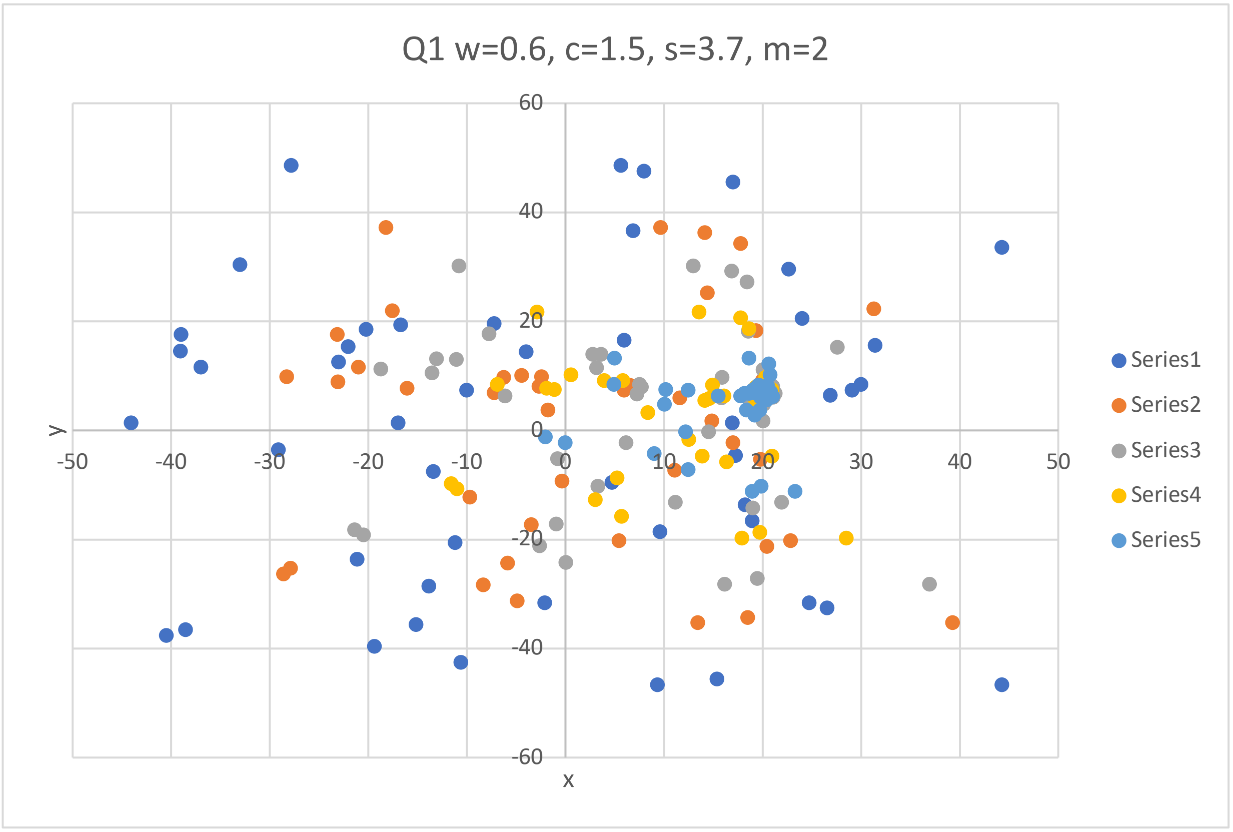

Figure 18. Particles with a large gap between social and cognitive parameters

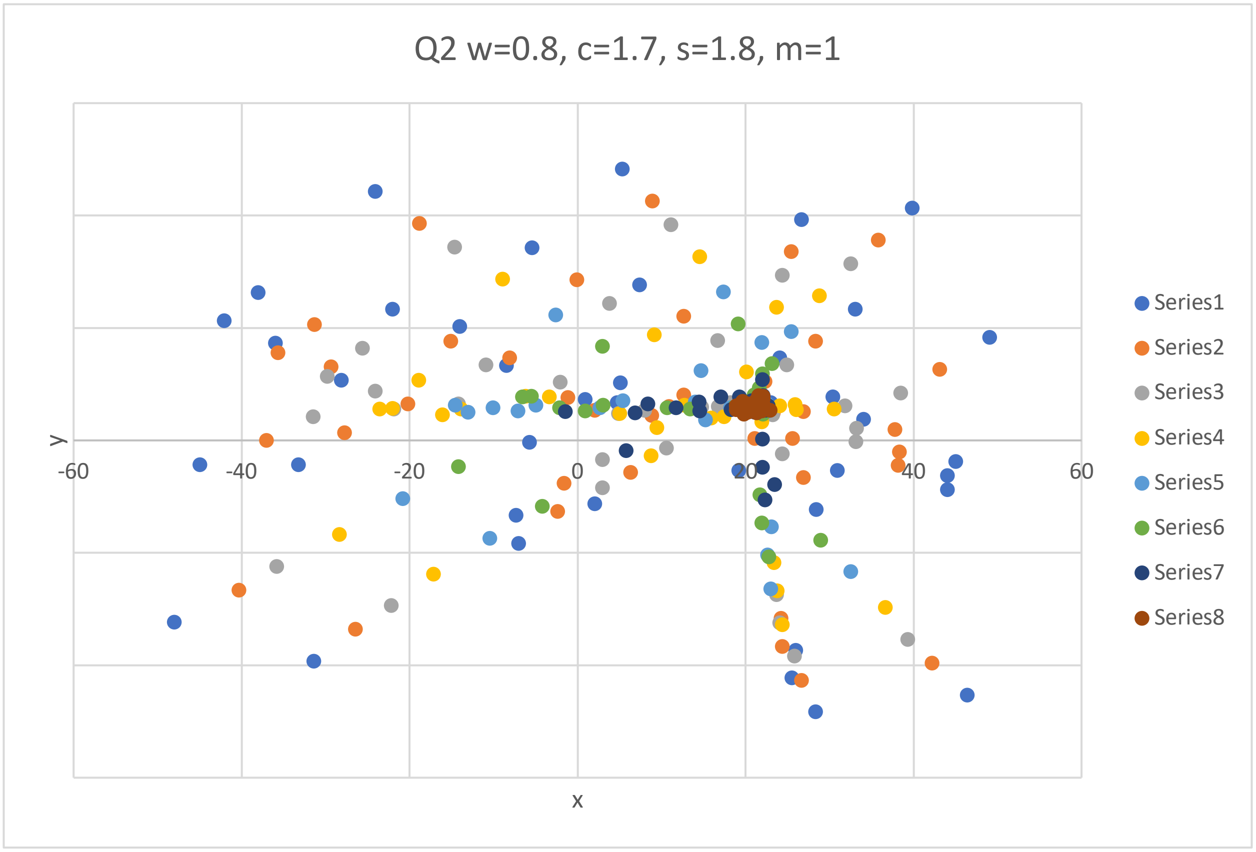

Figure 19. Particles with Q2 function

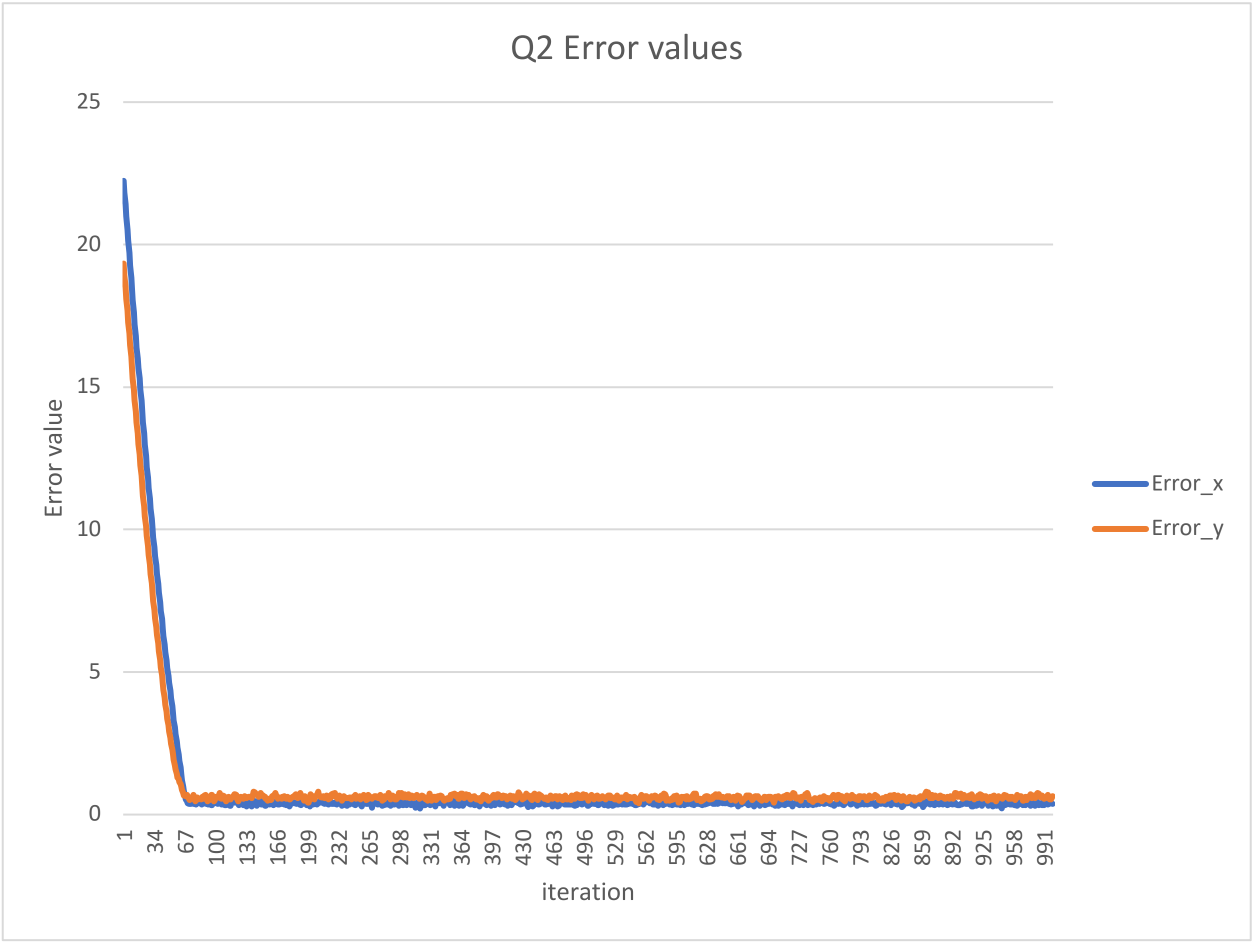

Figure 20. Error values with Q2 function being used

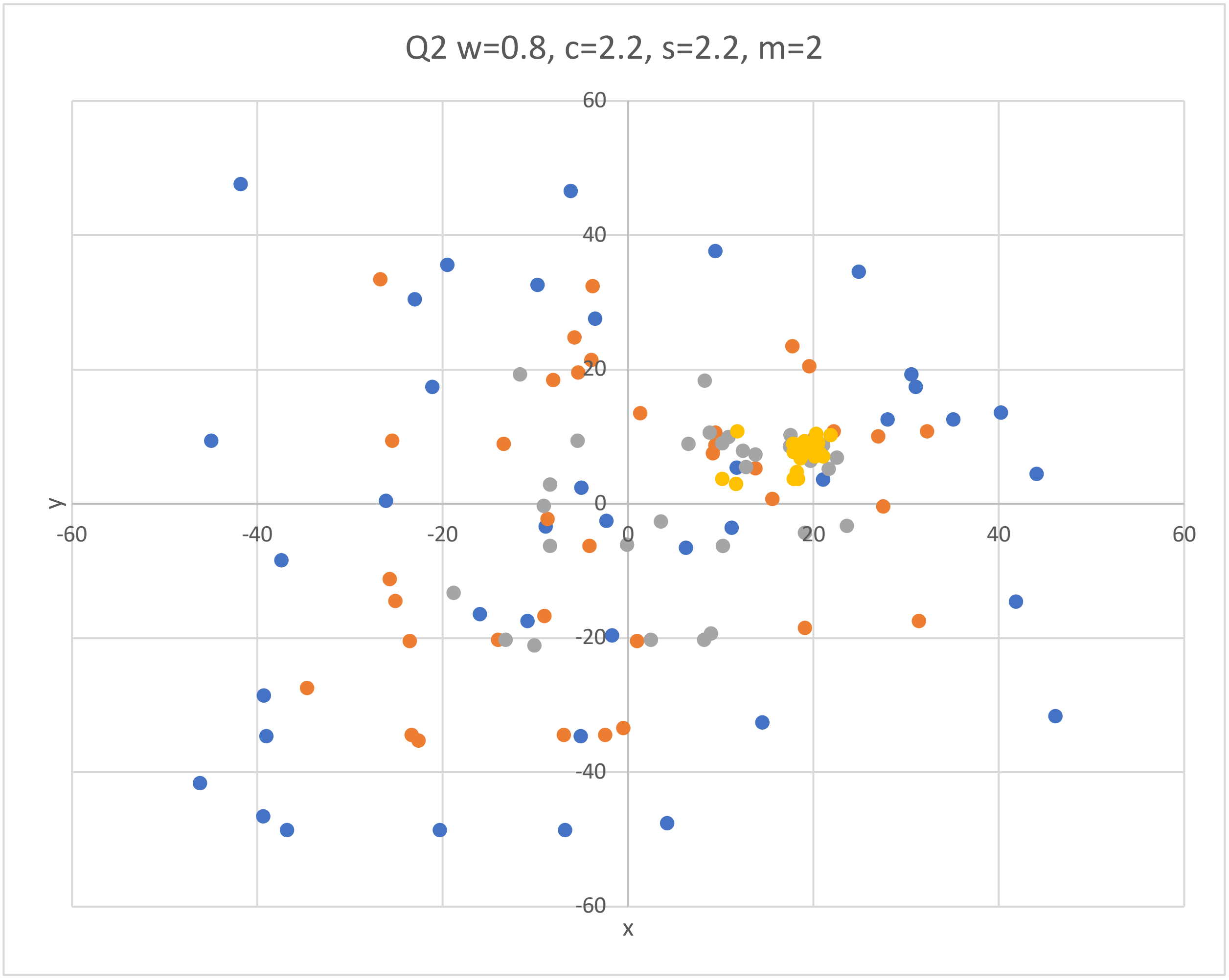

Figure 21. Particles when social and cognitive parameter were set equal to each other

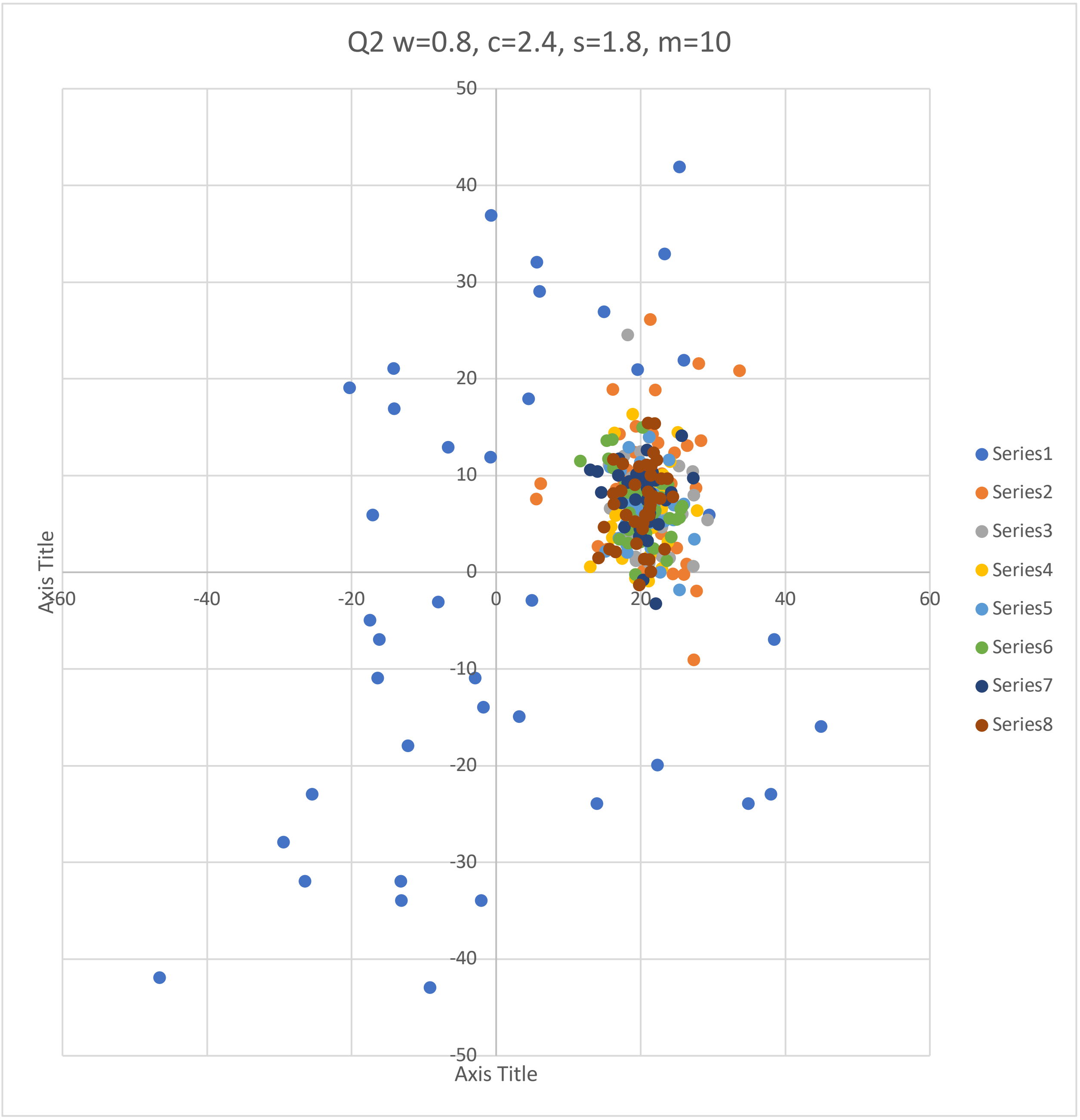

Figure 22. Particles when max velocity was largely increased

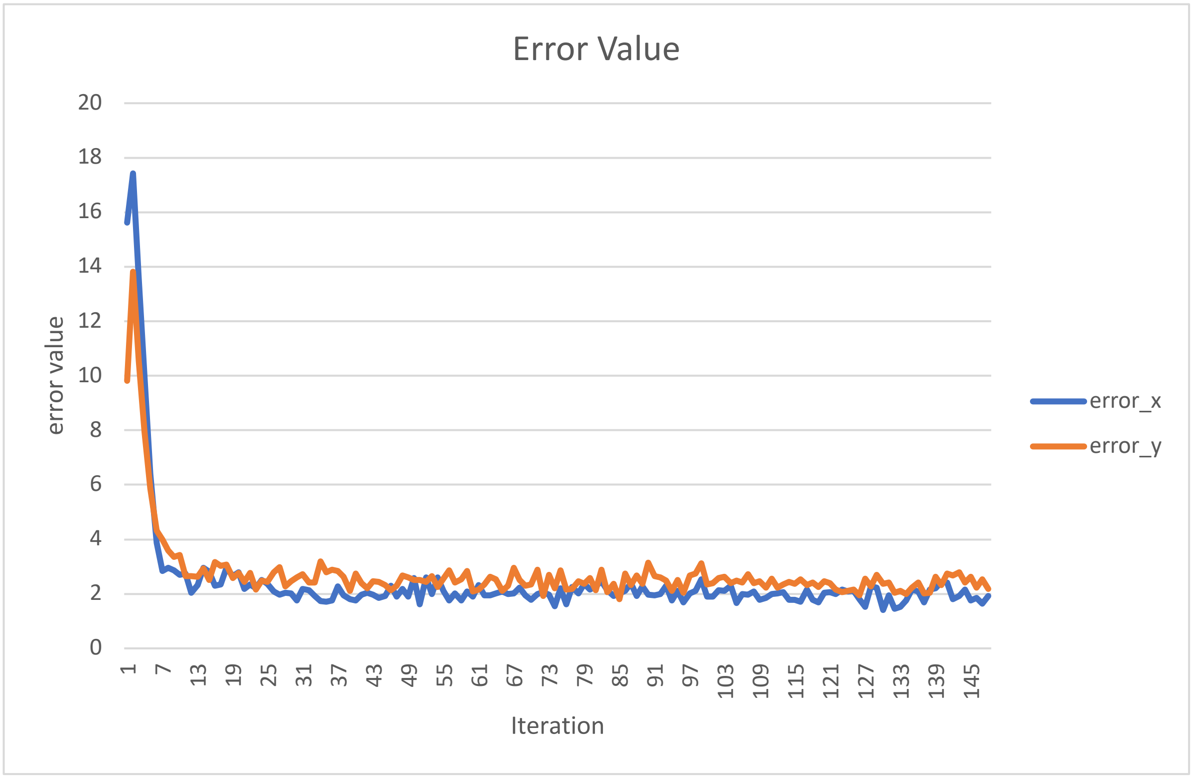

Figure 23. Error values for Q2 function and maximum velocity set to be large.

There were interesting things to see when conducting these experiments on particle swarms. When the Q1 function was used the particles converged to the maximum, (20, 7), every single time, an example of which can be seen in figure 11. The error values also steadily decreased until it got to just above 0.2. Once the value got to these lower values since, they were converging, only minor changes were needed to get to the goal point, this can be seen in figure 12. In figure 13 the social parameter was increased, and this caused the convergence to take slightly longer than the original, when the max velocity was increased convergence took less time as can be seen in figure 14. The error values shown in figure 15 is when the max velocity and social parameter was changed, this shows that the error values decreased at a faster rate. When max velocity was largely increased (to value 10), convergence took very little time, this can be seen in figures 16 and 17. In figure 18 you can see that when there is a larger gap between the social and cognitive parameters it takes longer to converge. In a general sense, it seems like when the social parameter is higher than cognitive the particles learn more from their neighbors and can come up with an optimal solution faster when they work together. When utilizing the Q2 function there was a general convergence towards the global maximum at (20,7), but, the local maximum of (-20, -7) seemed to not be of importance to the particles. As can be seen in figure 19, throughout the iterations you can see a clear path some of the particles took to get to where they ended up. Looking at the graph when particles got around the -20 x value, there seemed to be a bit of a cluster forming but then they eventually went and moved closer to the global maximum. Also using the Q2 function caused the population to take longer to converge, possibly because there were two optimal points vying for attention. Looking at the error plots in figure 20, the values had a steep decline and once the values reached around 0.4-0.5 the values continually fluctuated and never really got below the 0.01 value. In figure 21 the population converged after some time but, there is no clear path when both the social and cognitive parameters have the same value. The particles travel was sporadic but did converge over time. When max velocity was made to be 10, the population seemed to converge very quickly, they never all reached the maximum point though, this can be seen in figure 22. They seemed to hang out around the global maximum and sometimes they would move away from it only to move back towards it again. The error values regarding figure 22 are shown in figure 23. First the values got worse after the first iteration and then trended negatively and reached a fluctuation point rather quickly. These fluctuations were larger due to the maximum velocity being 10. This population never converged.

Conclusion

PSO was a great way to understand how the social behavior of some animals may seem weird to us actually have good usages. From understanding the process, a PSO problem can take, it was interesting to see how the parameters could ultimately affect the outcome of the population. Knowing these things can help you appreciate what it takes for groups of animals to do this. The benefits of using PSO to search a large area with many particles to find the best optimal solution gave good experience in how the swarm can be through of as either exploratory behavior, or exploitative behavior.