**Question Description**

The first question is a six lump camel back equation.

The second question is a sphere function.

**Algorithm Description**

The algorithm used here is a simple PSO technique. A PSO algorithm simulates a swarm of particles moving toward the solution. The x and y variables in the equation given in problem 1 are used as a position vector for each particle. The neighborhood best and global best position vector for each particle is the position vector that results in the lowest value from the evaluation function. The velocity vector is calculated by subtracting a particle’s current position vector by its personal best position vector and neighborhood best position vector, then adding the previous weighted velocity vector. After getting the new velocity, the position vector is updated.

After running the PSO algorithm 10 iterations, 100 iterations, and 1000 iterations, you can see the solution converges really quickly to -1.03163, in about 100 iterations. In addition, the solution converges slightly at a different rate with different parameter. In general, the solution converges and the average positioning of each particle varies greatly in the first 100 iteration. After reaching 1000 iterations, the average positioning also converges and stabilize to the solution.

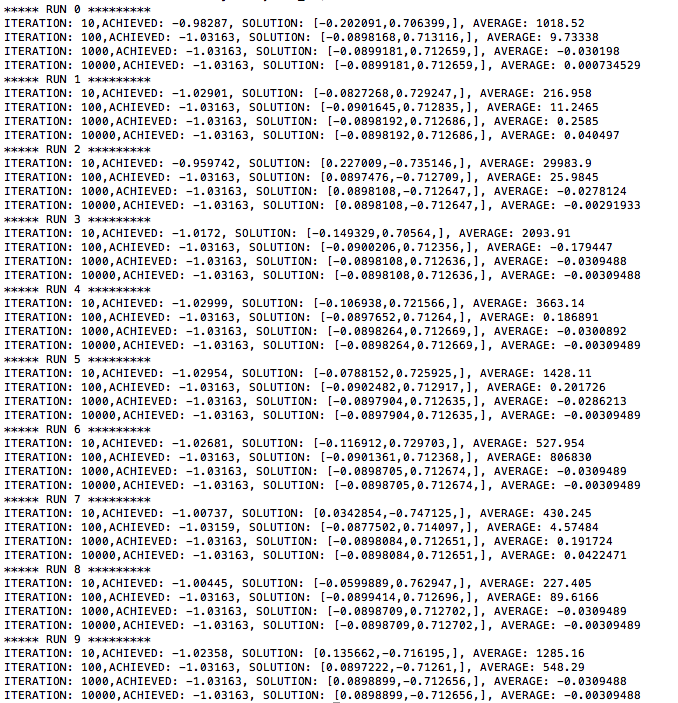
**Inertia weight version**

The inertia weighted version of PSO algorithm is really good with for exploring the different solution domains and converging quickly. Usually, within 10 iterations, the algorithm is very close to the solution. After the 100 iterations, the algorithm will stabilize at the solution.

Weight = 0.792

Personal velocity weight = 1.4944

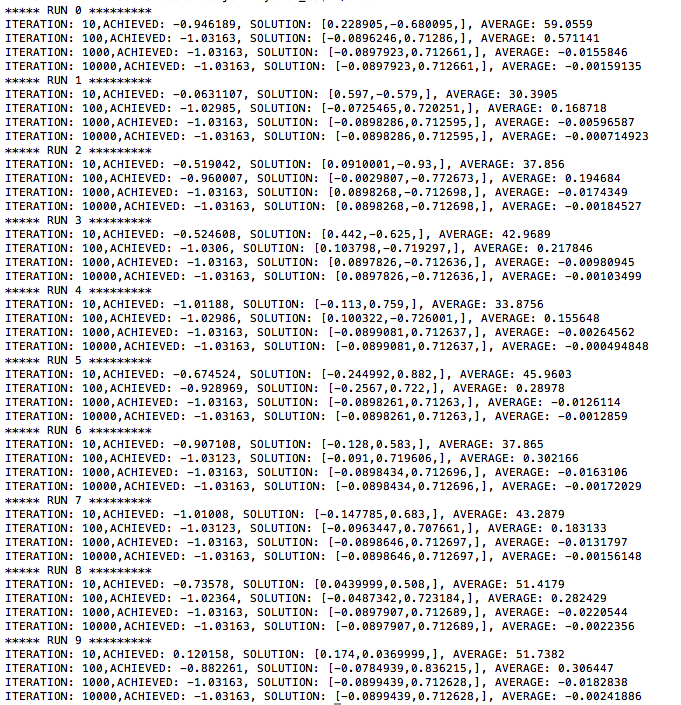
Neighborhood velocity weight = 1.4944



**Vmax Version**

The Vmax version reaches the solution a bit slower than the weighted version of PSO. It will slow down exploration to prevent the algorithm from going too far away from the solution, and speed up convergence. This can be seen from the solution of the PSO algorithm after 10 iterations, as it is worse than weighted version. In exchange, the solution converges faster.

Vmax = 0.01 for both x and y

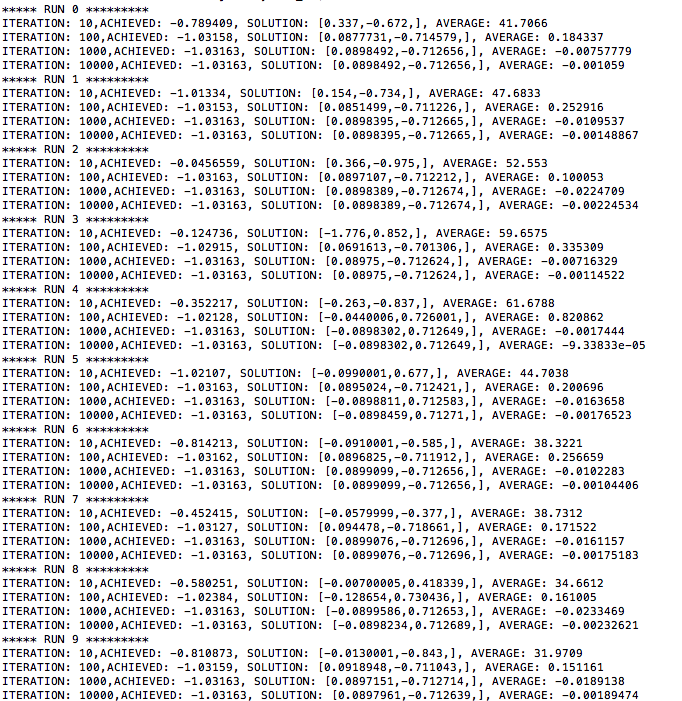


**Constriction Factor Version**

The constriction factor version converges a bit slower to than weighted version of PSO. Both the averages and solution converges only after 1000 iterations, whereas the previous 2 versions converge after 100 iterations. However, it guarantees convergence. Constriction factor requires that theta, the sum of personal velocity weight and neighborhood velocity weight, to be at least 4 to guarantee convergence.

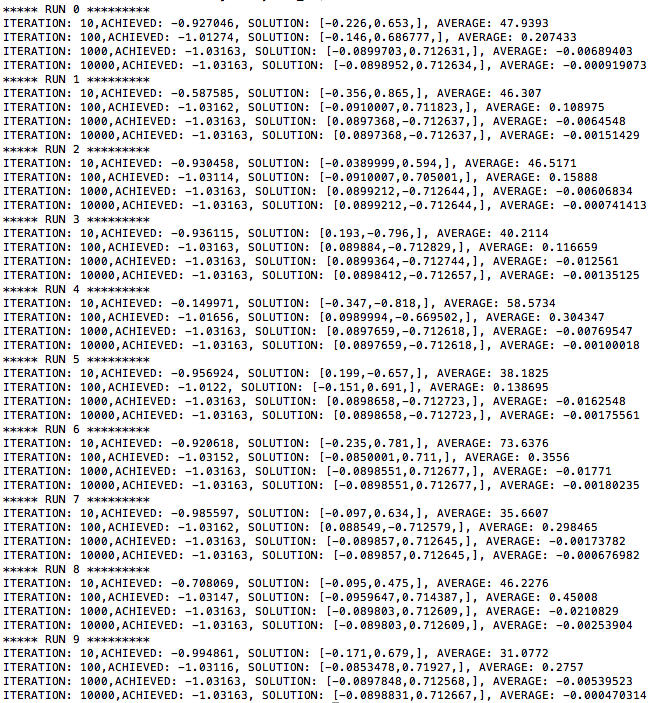
Personal velocity weight = 2

Neighborhood velocity weight = 2



**Neighborhood Size of Three**

Having a smaller neighborhood size will slow down the convergence significantly. This can be seen from the results of the PSO algorithm below. The solution only converges after 1000 iterations, and the average is not where near the solution yet.



**Changing Random Seed 10 times**

Changing the random seed did not seem to make any changes to the run results. The effect of clearing the random seeds depends on the programming languages used for this assignment. In this case, changing the random seed C++ did not result in any significant changes with the program.

### Problem 2)

**Algorithm Description**

The PSO algorithm used in the second problem is very similar to the problem above. It still updates the position vector by adding the new velocity by the old position vector. Hopefully, after iteration, the particles will get closer to the solution. The main difference between the algorithm implemented in problem one and problem two is the change in evaluation function, and the number of variables. Problem two requires 10 variables, whereas problem one requires only 2 variables. Thus, the position vector, velocity vector, personal best vector, and neighborhood best vector are set to an array size of 10. Secondly, the evaluation function in problem 2 is implemented as the sum of all variables’ squared.

The convergence of this problem is much slower than the previous problem. Convergences only happen after 1000 iterations, and sometime it may not each reach the solution, although, the algorithm gets fairly close to the solution after 1000 iterations. This makes sense, since this problem has 10 variables that need to be optimized, whereas the previous problem only has 2 variables.

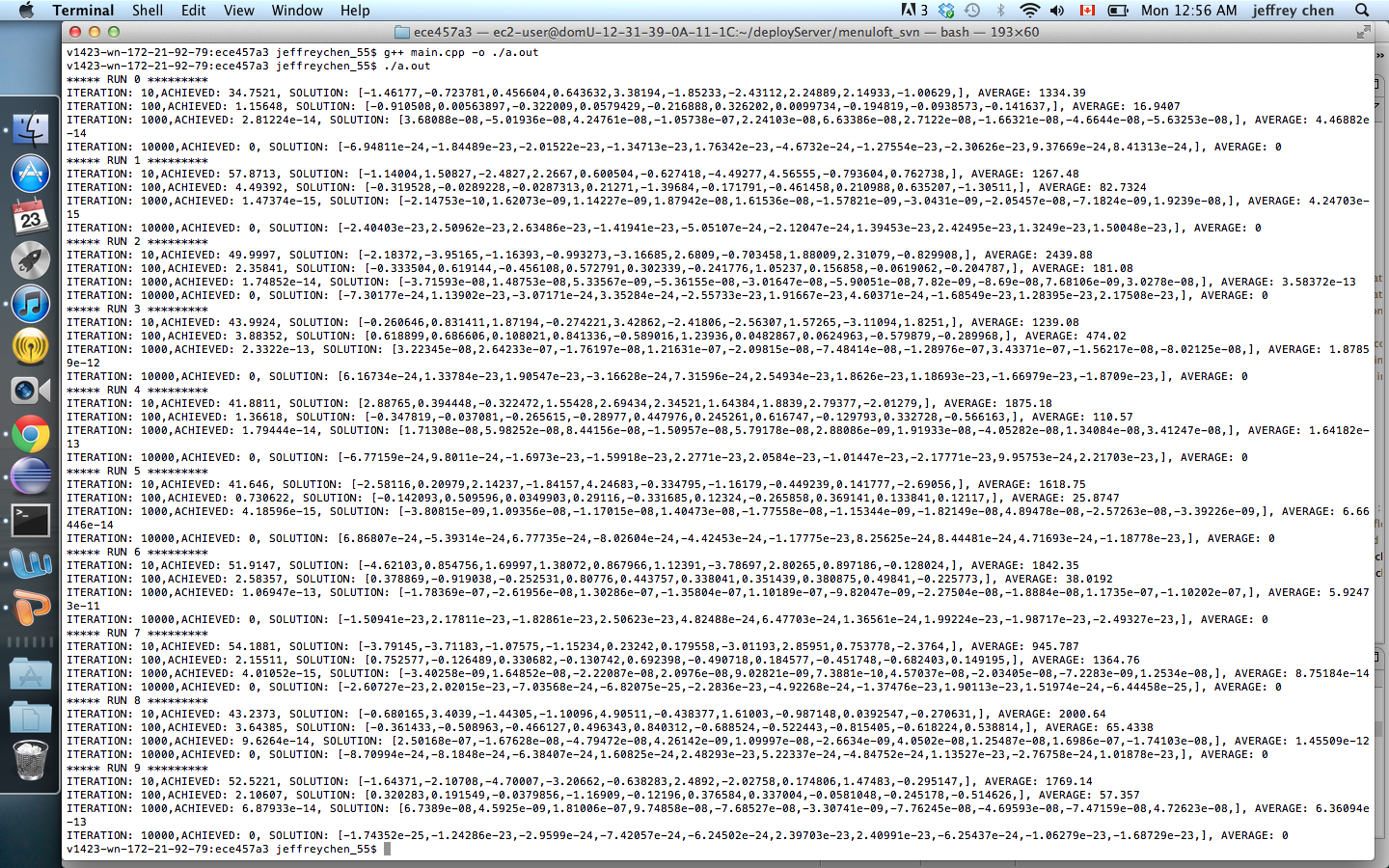
The different versions of PSO algorithm performs similarly to the previous question, except that it is significantly slower.

**Inertia weight version**

Weight = 0.792

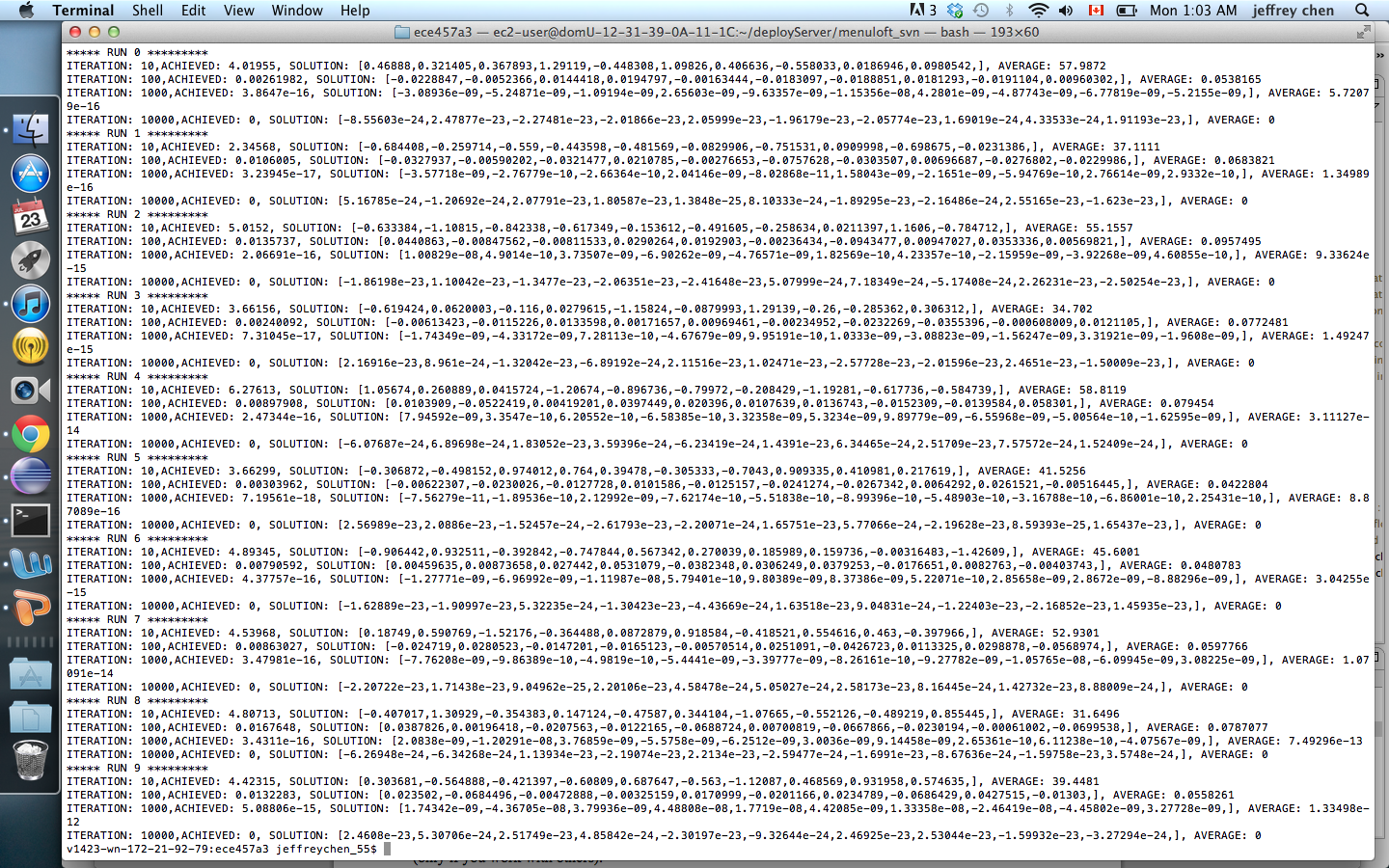
Personal velocity weight = 1.4944

Neighborhood velocity weight = 1.4944



**Vmax Version**

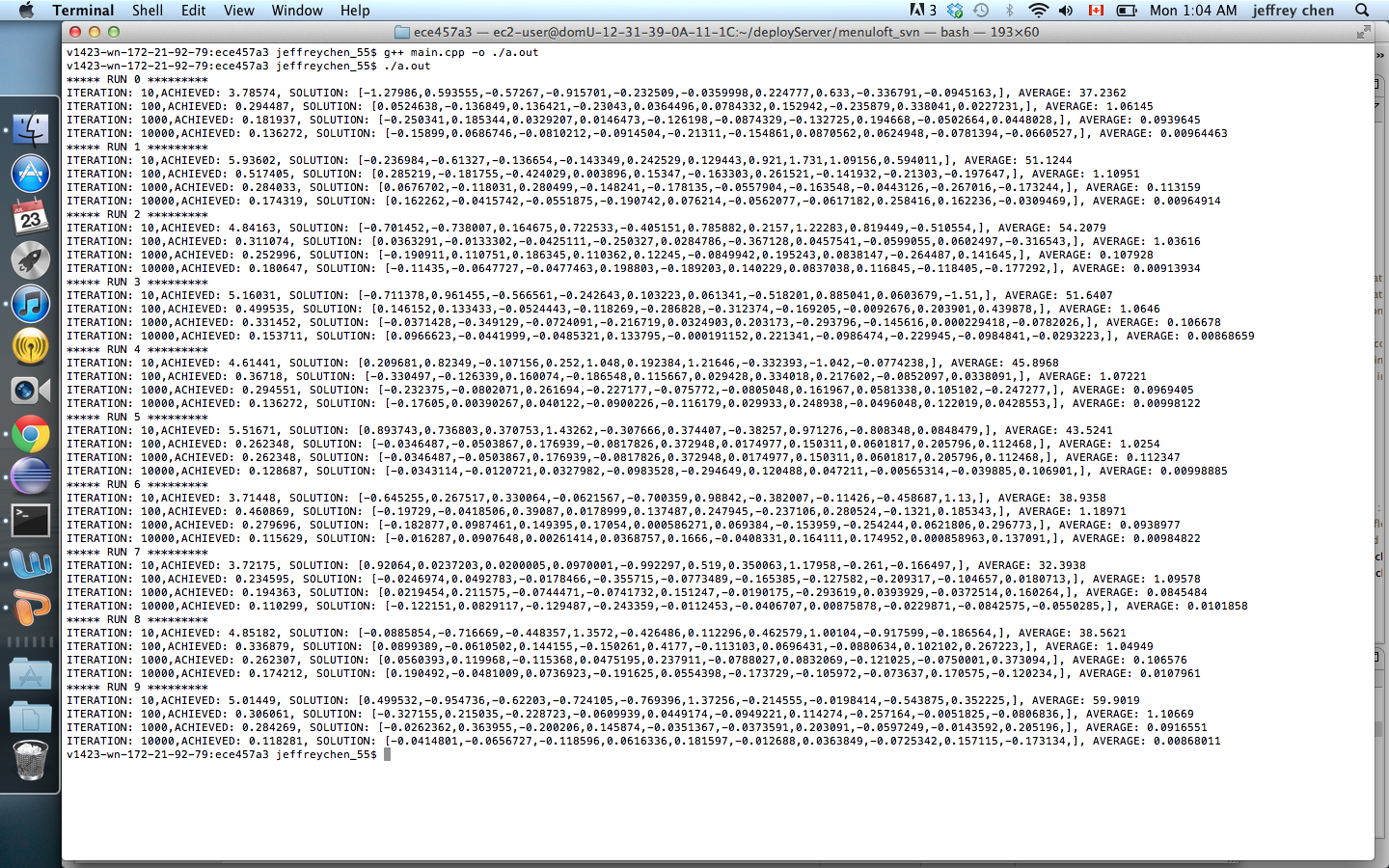
Vmax = 1 for both x and y



**Constriction Factor Version**

Personal velocity weight = 2

Neighborhood velocity weight = 2



**Neighborhood Size of Three**

