Particle Swarm Optimization for Minimizing the Rosenbrock Function

Bryan Tran

Abstract

In this study Particle Swarm Optimization with a global neighbor implementation is used to find the global minimum of the Rosenbrock function. Performance is measured as the inertial constant, population size, and maximum velocity parameters are independently varied. Results show a lower C0 constant and a population size of at least 20 produce effective results. Max velocity variation shows no discernible trend of effect.

$$f(x,y) = (a-x)^2 + b(y-x^2)^2$$

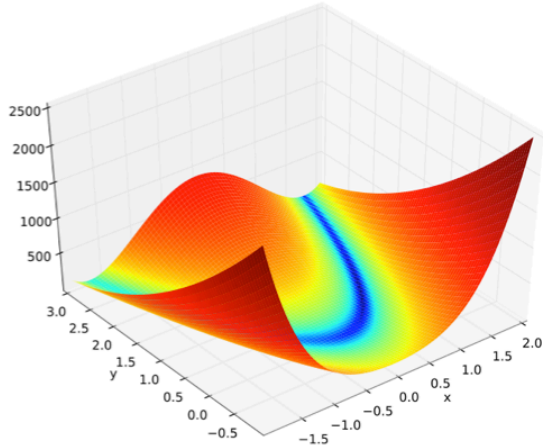

Figure 1

Methods

The Rosenbrock function, shown in figure 1, is used to measure the performance of the implementation of PSO. See (Xiaohui, 2006) for an overview of the PSO algorithm. In this study a=1, b=100, and the input variables are bounded by: -1.5<= x <= 2.0 and -0.5 <= y <= 3.0. Here, a global, social neighbor implementation is used where each particle's neighbor set is the entire set of particles. Consequently there exists only one global best at any time in PSO's execution. After initialization, the update phase of PSO is repeated until either convergence occurs, or the maximum number of iterations has been reached. Convergence is defined as the sum squared error (SSE) of the global best position (gBest) falling below the error criterion. It is known that the global minimum of the Rosenbrock function used in this instance is (1,1), so error is defined as the SSE of the global best position (compared against (1,1)) falling

below the chosen error criterion of 0.001. The maximum number of iterations is chosen to be 10,000.

Results / Discussion

In PSO a good number of parameters can be varied, and thus the effectiveness of the implementation can vary significantly depending on these parameters. A series of tests varying these parameters independently was used to 'fine tune' the system. The inertial constant, C0, was found to significantly affect system effectiveness. Recall that in the update phase of PSO, each particle's velocity and position is updated according to the following equations:

V[] = **C0** * v[] + c1 * rand() * (pbest[] - p[]) + c2 * rand() * (gbest[] - p[])

P[] = p[] + v[]

Where **C0**, **V[]**, **P[]**, are the inertial constant, new velocity, and new position respectively (Xiaohui, 2006). As figure 2 shows, SSE of the system rises significantly once C0 exceeds 0.2. With population size, as figure 3 shows, performance improves considerably once size is increased towards 20. Past 20, however, performance boost plateaus strongly. Outlying data points are attributed to noise, as the system was run only once with each population size. As opposed to size and the inertial constant, varying the maximum velocity allowed showed no trend in affecting the system. See figure 4 to see SSE as max velocity is increased. Lastly, the system was repeatedly executed with a fixed set of parameters, with (C0 = 0.2, C1 = 1, C2 = 3, size = 20, and max velocity = 0.01. Repeated executions showed a success rate of approximately 80%.

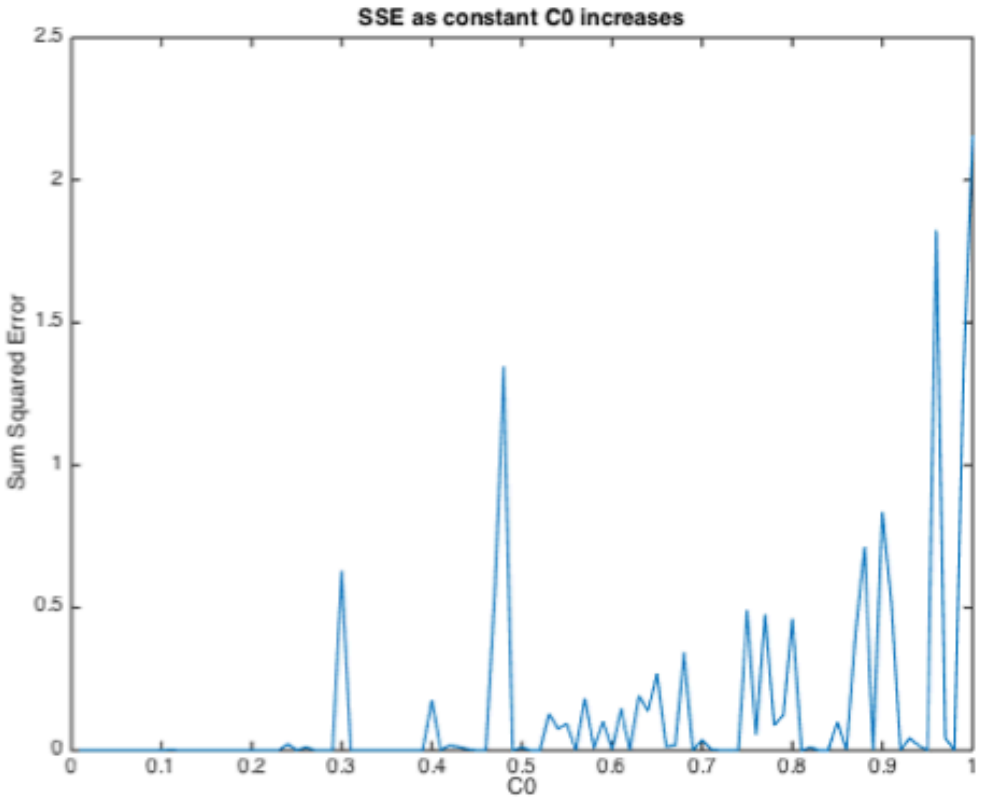
SSE as constant C0 increases
Sum Squared Error
C0
2.5
2
1.5
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Figure 2

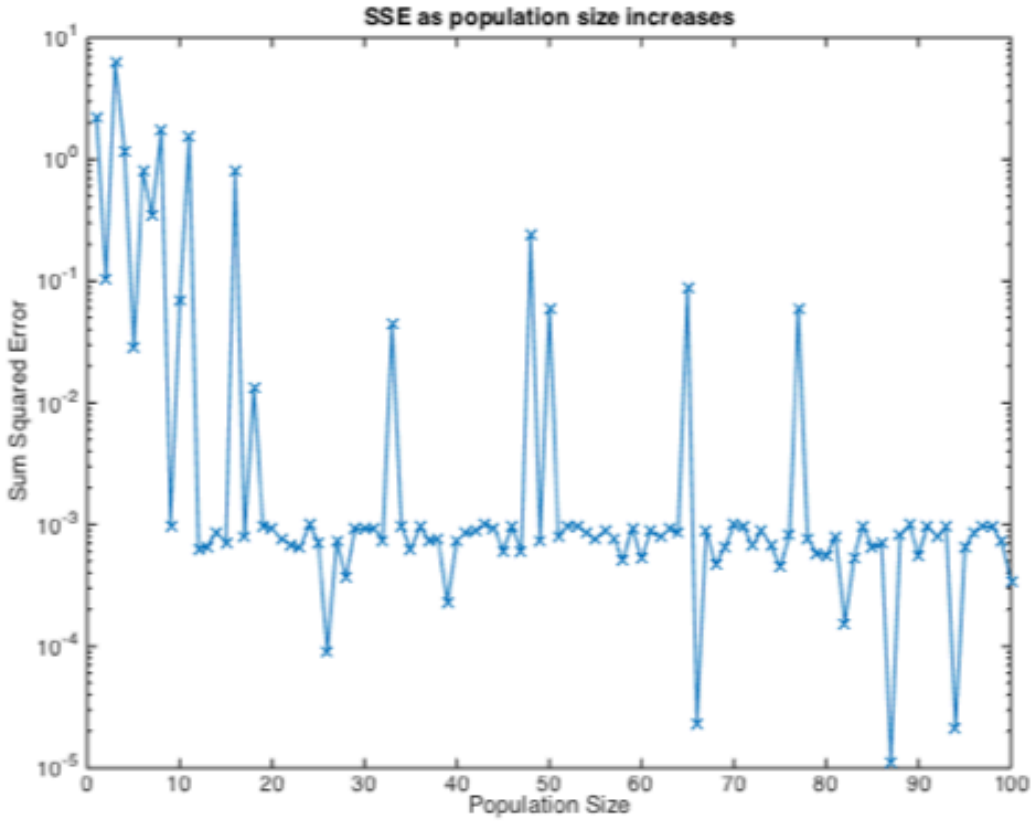
SSE as population size increases
Sum Squared Error
Population Size
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
10
20
30
40
50
60
70
80
90
100

Figure 3

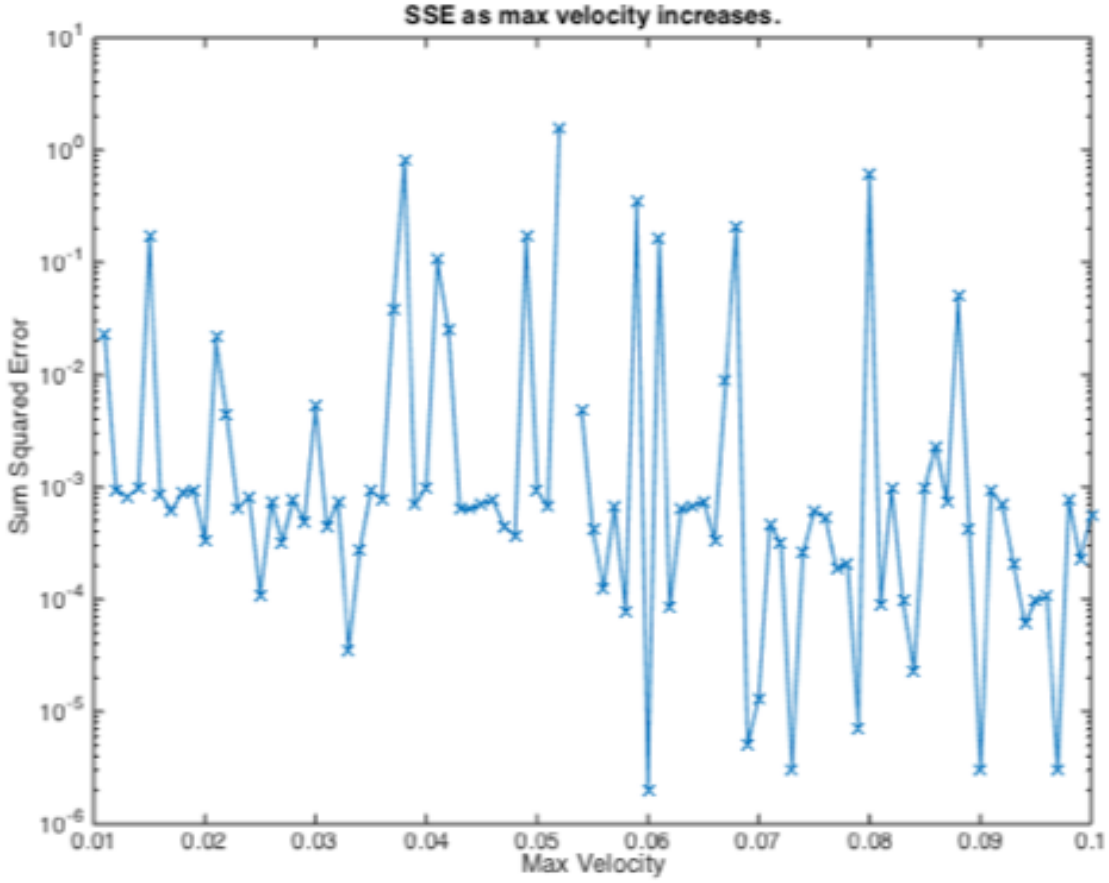
SSE as max velocity increases.
Sum Squared Error
Max Velocity
10^1
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
10^-6
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

Figure 4

Since the problem space is rather small, it is not unexpected that a lower inertial constant produces better results. Once C0 is increased beyond 0.5, the particles have a tendency to become too “headstrong”, ignoring global information about the best location. The author suspects the problem space is best suited for group thinking, heavily penalizing individual inertia over tendencies of following the group. The author was, however, surprised to find that varying maximum velocity within the range 0.01 to 0.1 had no general effect. Outside of this range the effectiveness of the system quickly degraded. This is to be expected, as a velocity value too low results in long convergence times, but a too high velocity value results in the particles overshooting the global minima. The author suspects that within the tested range, the other parameters are simply more influential than velocity in affecting output.

Conclusion

A PSO implementation for minimizing the Rosenbrock function was presented and evaluated. A variety of parameters were tested individually, with the author concluding that a lower C0 constant, population size of at least 20, and a bounded max velocity produce effective results in finding the global minima. Future development should implement different neighboring schemes, such as restricting the number of neighbors, or a distance-based geographical neighboring scheme.

References

Rosenbrock function. (n.d.). Retrieved May 04, 2016, from https://en.wikipedia.org/wiki/Rosenbrock_function

Xiaohui, H. (2006). PSO Tutorial. Retrieved May 04, 2016, from http://www.swarmintelligence.org/tutorials.php