Brian Burton
CS 2420
03/23/17

Analysis Document

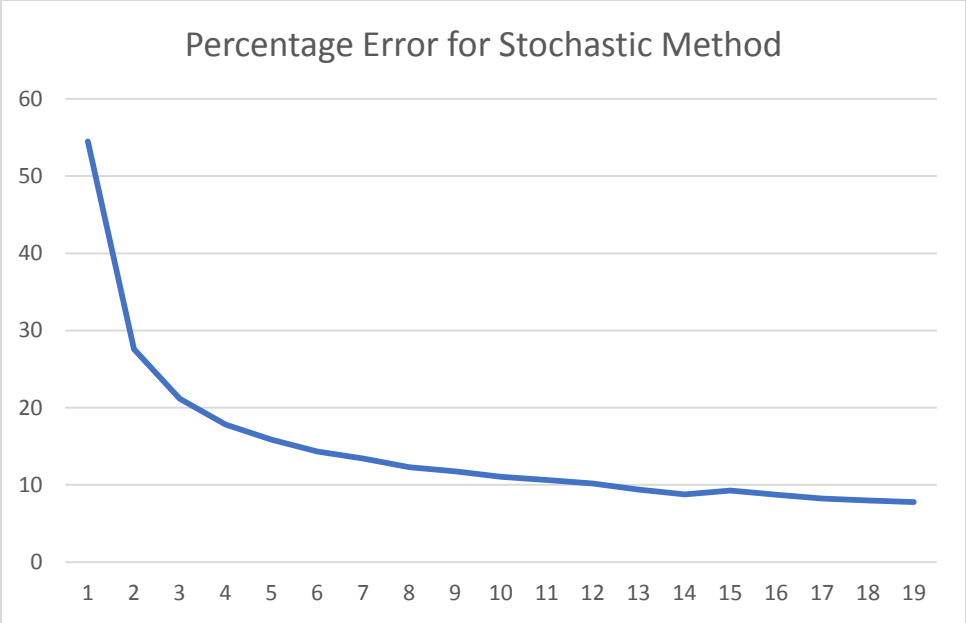
1. Overview of Project

What I was able to learn in this assignment about a stochastic processes is that you must have a way to generate a random number. Without a random number you will just get the same answer or outcome. This will cause you to receive false results. I also learned very quickly that the stochastic process is a lot faster than an exhaustive process. This is because the stochastic method allows you not to have to go through and look at every possibility. It is able to approximate quite well instead. Though I will say that exhaustive methods are easier to write.

What I learned about random number generation is that it is basically impossible to truly create a random number on a computer. Not only that but it is super hard to even try to get a random number. With the right numbers and functions you can come close but it will never be perfect because computers are so concise. If you give a computer an input, it will always give you the same output. This fact makes it nearly impossible to truly create a random number generator.

2. Timing and Graphs

Show the Accuracy vs. Sampling Rate



The trade off in time versus accuracy is dependent upon the number of samples you are observing. If you are evaluating a small number of objects, then using a exhaustive method does not cost you very much time and is extremely accurate. On the other hand, if you are evaluating a very large number, then a Stochastic approach is much better for two reasons. First, the more numbers you are evaluating the better the accuracy, and second, it takes significantly less time to compile. I believe that a good trade off would be somewhere around 1 million objects.

I would expect it to take 8 minuets to compile an exhaustive method that evaluated 9 cards. I believe this because it took about 2 minuets to evaluate 7 cards. I also believe that because of the Math behind it. To evaluate 5 cards, that is fifty two choose five. That would equal 2,598,960 possible hands. If you evaluate 7 cards, it would equal 133,784,560. 9 cards would be 3,679,075,400.

3. Software Development Log

Lucas and I spend about twelve hours on this assignment. The most time consuming part of this assignment was finding what the rank of a hand of cards was. It is still not perfect unfortunately but it is close we believe. What I can do better in the future is I can better understand the directions of the assignments. I believe that if we would have done this then we would have found a much better solution for this problem. I believe that we had allocated enough time in order to finish on time. I always wish that I had more time however to make the program more efficient and understandable. It just isn't always possible to have that kind of time. Unfortunately we started writing tests half way through the assignment. After we

started using the testing class however, I believe that our progress on the assignment was much faster.

4. Texas Hold'em Analysis

The insights that I gained from the Texas Hold’em empirical analysis was never to gabble. The odds will never be in your favor. There are just too many chances for you to loose. In addition to that, I learned that the top ten best 2 card hands that you can be dealt in Texas Hold’em is two aces. The rest going from top to bottom are two kings, two queens, two pairs below jacks, and everything else. Two Ace cards beat every other combination of cards almost every single time. We used one billion samples in order to find a valid probability of winning. The reason that we used this number was because we figured that one billion samples is the largest amount of samples that we were willing to wait for our program to run.

5. Thought Problems

A random number generator is extremely important to a stochastic experiment because without it you will receive a super inaccurate approximation. So since stochastic methods are meant to make good approximations, it is essential to have a random number generator. Our worst generator method did not compare to Java’s since all it could do was return one number. Our best generator method compared a lot better to Java’s. There are a lot of problems still. Like for instance there were no odd odd or even even pairs. However, our average number was fairly close and our median, min, and max number of times a number was generated was dead on to Java’s. I think that another good test for the Check Random Class would be to check for how often each number is used or at least the number that is used the most. I believe that that could help the developer determine whether or not his or her Random Generator is using a curtain number too often. The only difference that I can see when using our random number generator and Java’s on our Stochastic method is the total percent error. When we use Java’s generator, the Stochastic method seems to be more accurate. When we use our generator, we follow the same path as Java only a worst approximation.