**Project 3: Self Avoiding Random Walk Beau G.**

**Partner: Joe Kelly Date: 4/3/2019**

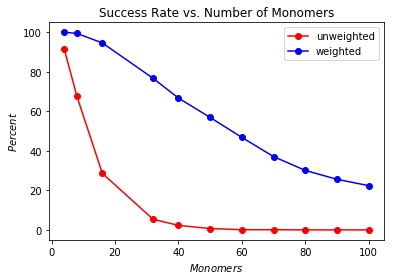
**Introduction**

The goal of this project was to apply random walk simulations to the idea of modeling the average length of a polymer chains. In modelling the average length of the polymer chains we were tasked to implement two different methods of self-avoiding random walks and analyze the results between the two. The two methods that will be implemented and compared are unweighted versus weighted self-avoiding random walks. The methods will be compared by taking important information from the walks (mean length squared, percentage of successful walks) for differing monomer counts and creating plots.

**Part (A): Python Script for Random Walk (Non-Weighted)**

Our first objective was to create a program that simulated an unweighted random walk. The method is started by a movement up and the current movement is declared as the ‘forward’ movement. The program then determines a random number between one and three and depending on whether it is between zero and one, one and two, or two and three, the chain will either expand left, right or continue forward. All movements are determined from the current movement marked as ‘forward’. In the unweighted implementation the walk will be ended at any point that the desired movement is onto a location that has previously been visited. After the walk has been terminated, the distance squared is calculated using the distance formula and is then recorded in an array for plotting. We also calculated the percentage of successful walks for plotting. The python script for this program has been attached and the results will be discussed later in the report.

**Part (B): Dependence on ‘N’ and Max ‘N’ that can be considered**

Next we will discuss the functions dependence on ‘N’ (The amount of steps for a walk). The functions themselves are directly dependent on the value ‘N’ as higher ‘N’ values in the unweighted method will result in throwing out more walks because it will be more likely to walk back onto itself and in contrast smaller ‘N’ values will result in more successful walks because it will be less likely to walk back onto itself. So at a certain point for the unweighted method the odds of success will go toward zero as ‘N’ gets bigger resulting in a technically maximum ‘N’ value. In the weighted method the odds of success for larger ‘N’ values is much higher as the walk is way more efficient but it will still eventually hit a point where odds are success are practically nothing therefore also has a maximum ‘N’ that can be considered as well. With that in mind, we can create a plot of the percentage of successful walks for different ‘N’ values using both the weighted and unweighted values and observe at which point the percentage approaches zero, giving us our maximum ‘N’ values. The following plot shows the percentages of success versus ‘N’.

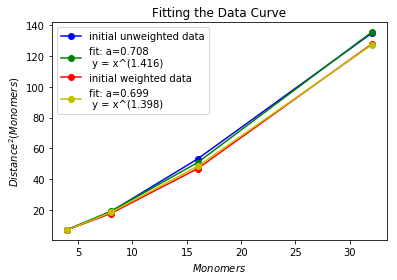
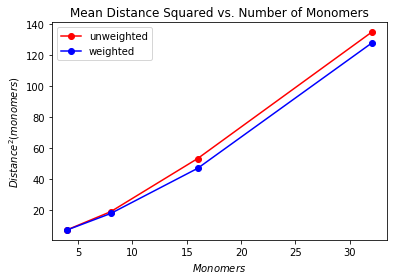
First we will analyze the behavior of the unweighted method. From the plot we can see that the unweighted method approaches a zero percent of success roughly around when ‘N’ equals forty. The weighted method is way more efficient in its walks therefore leading to a much higher rate of success than the unweighted. Even at an ‘N’ value of one-hundred the odds of success are still around twenty percent. Eventually the weighted method will approach a point where it will hit a maximum ‘N’ but in our case going above a hundred isn’t necessary so we can just state that its maximum ‘N’ is somewhere in the one-hundred plus range.

**Part (C): Python Script of Random Walk and Mean Distance Squared (Weighted)**

The weighted program is similar to the unweighted random walk in how the movements are chosen but with slight changes. Instead of ending the walk if it wants to move to a location that it has already visited, we can use weighting to continue the walk in a random direction chosen from the remaining viable movements from the walks current location. In order to get the correct mean distance squared with this implementation we needed append the distance squared and the weight after each walk to an array. After all walkers have completed their walks we were then able to find the use an average function to find the mean distance squared from the array. As in the prior method the mean distance squared and the percentage of success will be recorded for plotting. This method will be found with the unweighted method in the attached python script.

**Part (D): Fitting Power Law () to the Plot**

The main reason we saved the mean distance squared for each method for plotting was to show that both the weighted and unweighted methods result in roughly the same curve for mean distance squared versus ‘N’. We can show this by plotting the two methods results and finding the line of best fit for each. We were tasked in finding the line of best fit for the curves based on the equation , where we will solve for an ‘a’ for both lines and compare the two to see how the lines relate. The first of the following plots shows the mean distance squared versus ‘N’ for the two methods and the second plot shows the same results but with a line of best fit for each.



From the first plot it is clear that the two methods share a similar curve. The noise between the two graphs can be narrowed down to the randomness of the trials as the results can differ depending on the variables used. Most of the trials resulted in the trend of the first plot. In the second plot we can see the fits for the two curves and from the legend we can see the two ‘a’ values found in the fitting. The unweighted curve resulted in an ‘a’ of 0.708 whereas the weighted method resulted in an ‘a’ of 0.699. The ‘a’ values are very similar when taking the random noise into consideration therefore confirming the idea that the weighted and unweighted methods result in no major difference of mean distance squared. Just to note, if the use exceeds the maximum ‘N’ value for the unweighted and compares the results to the weighted method there will then be a difference between the two results as the amount of walks thrown out in the unweighted method will greatly affect the mean distance squared and in the weighted it will run normally because it can handle the larger ‘N’ value.

**Conclusion**

Overall the application of the random walk in the case of polymer chains was a success as both methods successfully simulated accurate representations of random polymer chains and the resulting plots displayed the relationship between the two methods that was expected.