Brandon Walters

Prof. Berzins

CS 3200

Homework 1 Report

The procedure I used for these problems was extremely similar, with only a few tweaks based on each specific problem. The general procedure was to start by creating 3 arrays, one for the n value of each specific iteration, one for the time the operation took, and one to contain the results of the model equation we were using. We then used a for loop to run each specific equation with an n value specific to that loop iteration. After storing all of the values needed, we would then loop back and go again. After the final loop iteration, we could then create the plots necessary. Figures 1 and 2 are the conventional and semilogy plots using our proper timing values, and Figures 3 and 4 are the conventional and semilogy plots for our model equation, a*(n/1000)^p.

1. For dot products of two random vectors, I found that I could get N up to around 10,000 and still have results in less than 30 seconds. However, trying to go up to 100,000 caused a freeze on my computer. I believe that you could achieve a reasonable time to return at N = 50,000 entries, but it definitely depends on hardware constraints. I did see the expected computational complexity, which was linear with O(N) complexity. The plotted equation model that I found to most closely resemble the plots generated by the operation itself had A = 50 and P = 1. This plot has a very close resemblance to the one generated by our model.
2. For matrix multiplication, once again I had the best results at N = 10,000, but there is definitely room to push that higher. My plots showed a linear complexity

of O(N) again, which was not what I had expected (O(N^3)). The plotted equation model that I found to most closely resemble the plots generated by the operation itself had A = 50 and P = 2. This plot has a very close resemblance to the one generated by our model.

3. For a system of linear equations, I found that I could get results up to N = 100,000 within 5 minutes. I have put it back down to N = 10,000 for stability reasons, as I had some crashes when trying to run Matlab, but it is certainly possible. The equation model for the last question partially applies, although I had to change the P-value from to 1.
4. Repeating our experiment with a modified identity matrix yielded similar results to the last question, although my N had to be lowered to 1000. However, the plots I got from this identity matrix seem to be more quadratic than linear. From the results I’ve had in this experiment and the previous one, it seems as there is some kind of low-end quadratic that Matlab uses to solve these systems. I was able to use the same equation model from the last question for this one, however, with a = 50 and p = 1.