**MOVING AVERAGES**

**LAB 8**

**SECTION X**

**SUBMITTED BY:**

**NATHAN SHULL**

**SUBMISSION DATE:**

**NOVEMBER 10, 2016**

**Problem**

The purpose of this lab, moving averages, was to read in data from an esplora, and create a program which reads input, and outputs data that is smoother, easier to read, and less “choppy.” In order to do this, I had to use what is called a moving average. Depeneding on the window size, the average would be taking for every 20 pieces of data, or every 100.

**Analysis**

The problem stated that I had to use the functions that were already in the skeleton code. Therefore, I was already given variables to use and didn’t have to create many. One thing that took me a while to understand was how exactly to calculate the moving average, and how to continually update it.

**Design**

The problem was to finish writing the functions that the skelton code gave me, so the basic outline of the program was already there for me. I first decided to work on the avg function, followed by the maxmin, and then the updatebuffer. The first two functions, for me were rather simple. They both just used a for loop that iterated through an array. Avg iterated through the buffer array and assigned that value plus the sum to the new sum. For maxmin, I iterated through the array titled array, and if it was bigger that the previous max, than that was the new value for max. And if the value was lower than the previous min, that was the new min.

The function that caused me some trouble was the updatebuffer function. I struggled trying to shift length-1 elements of the buffer to the left and put the new\_items on the right. After working with peers and a couple TA’s, I discovered how to do it. I also learned that I needed to assign new\_item to buffer[length-1], and NOT the other way around, as this didn’t update the max and min.

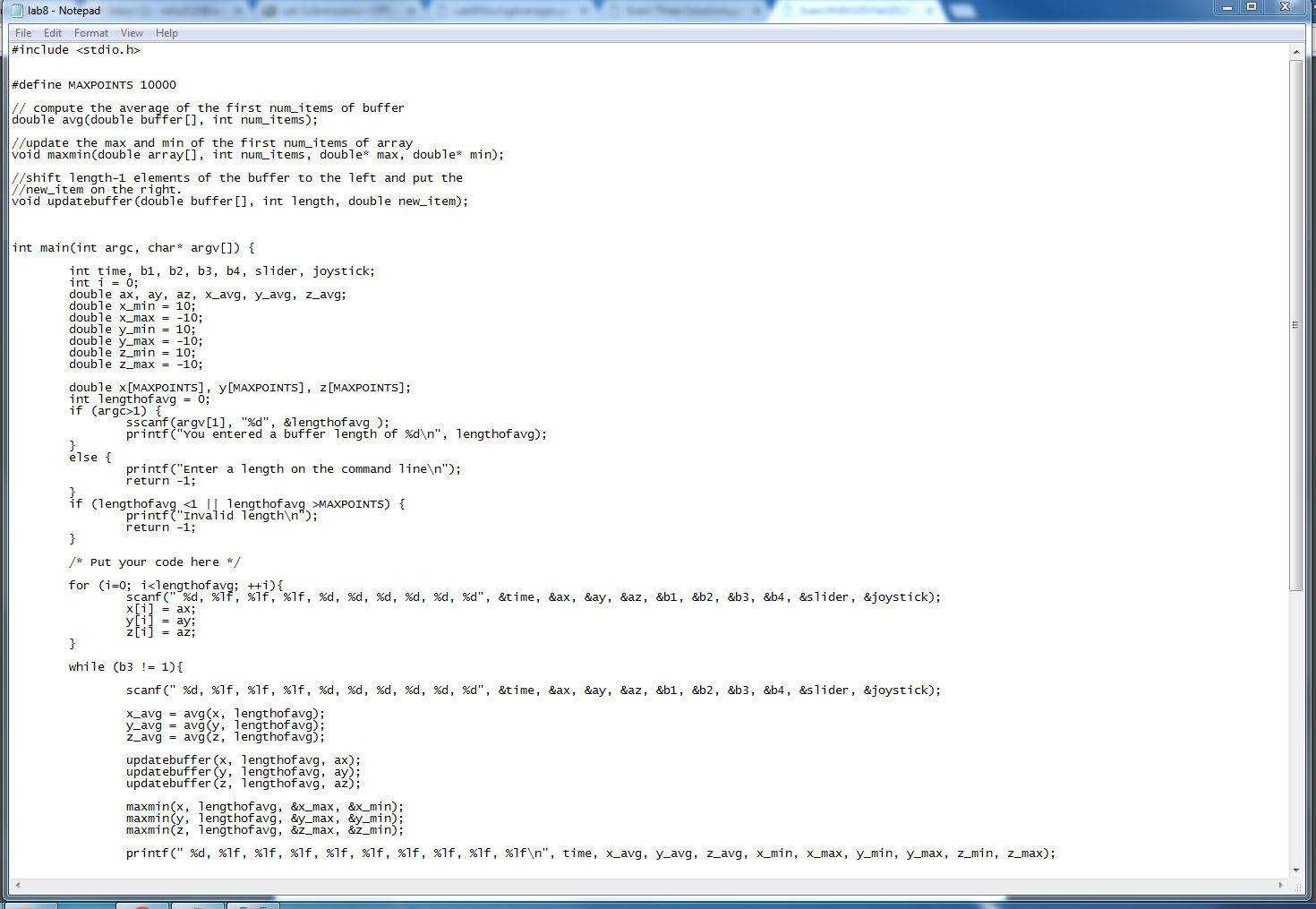
**Testing**

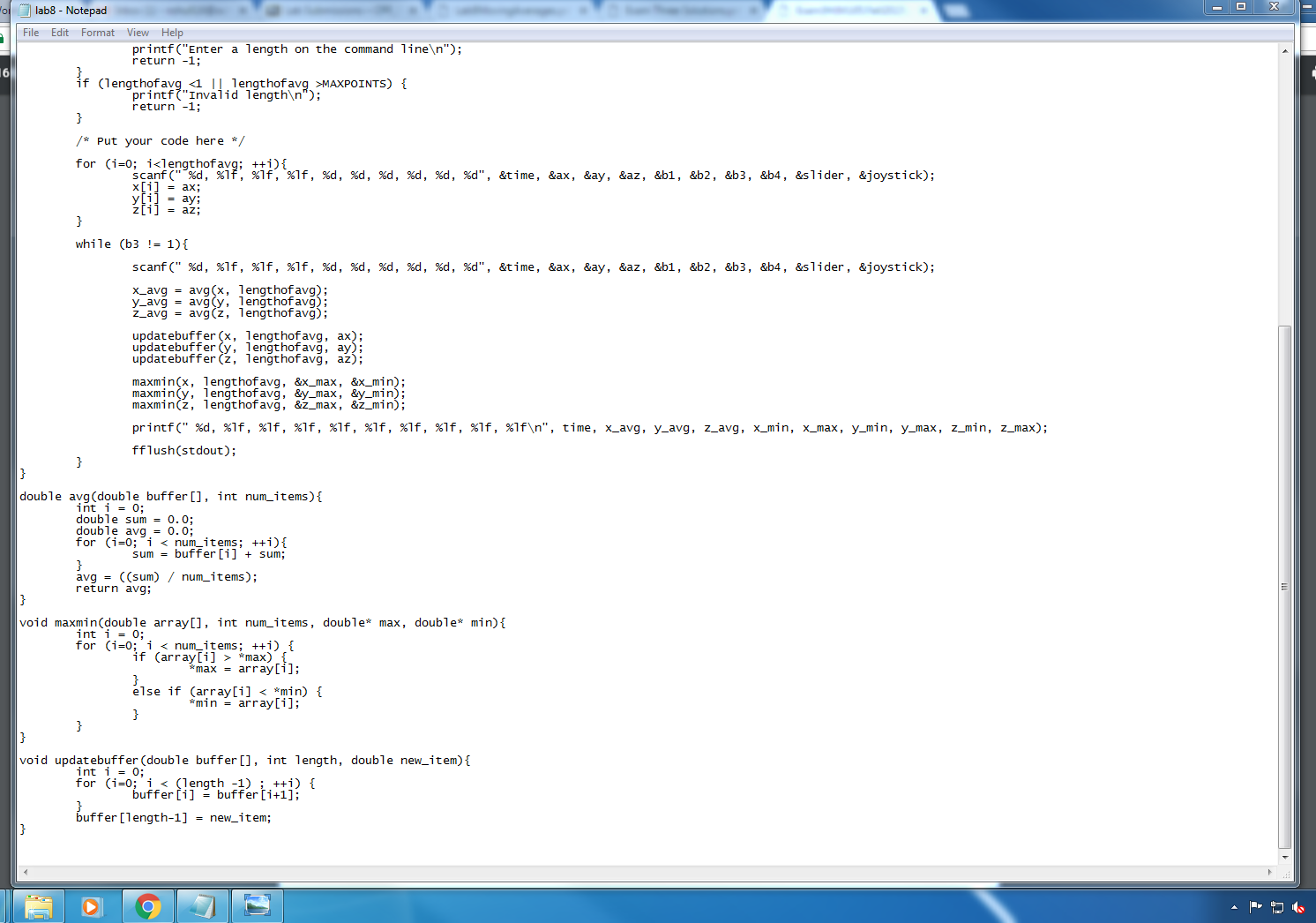
In order to verify the results of the solution, I had to use and esplora and look at the values it returned. On my first run through, all my values stayed the same after the first line. I realized this was because I didn’t have another scanf statement in my while loop. Another error that caused some confusion in my program was how I ended my updatebuffer function. Originally I had new\_item = buffer[length -1]. I later fixed it to buffer[length-1] = new\_item. It made me aware of how important the order is when assigning on number to a variable.

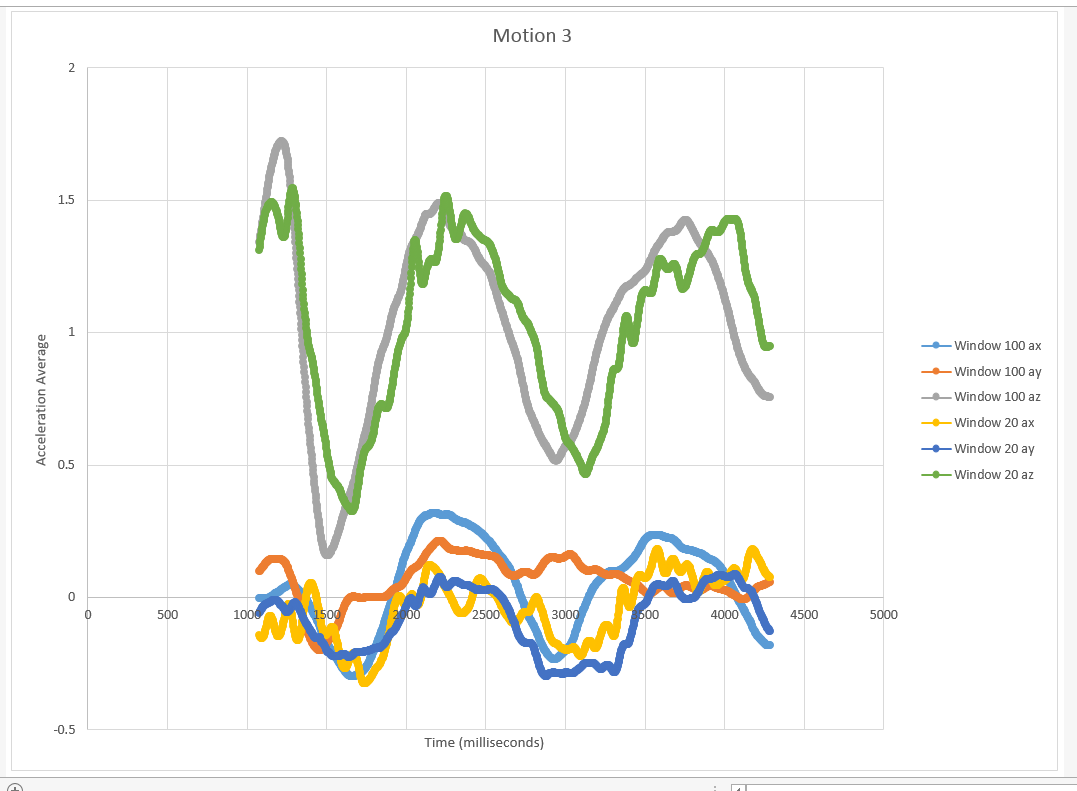
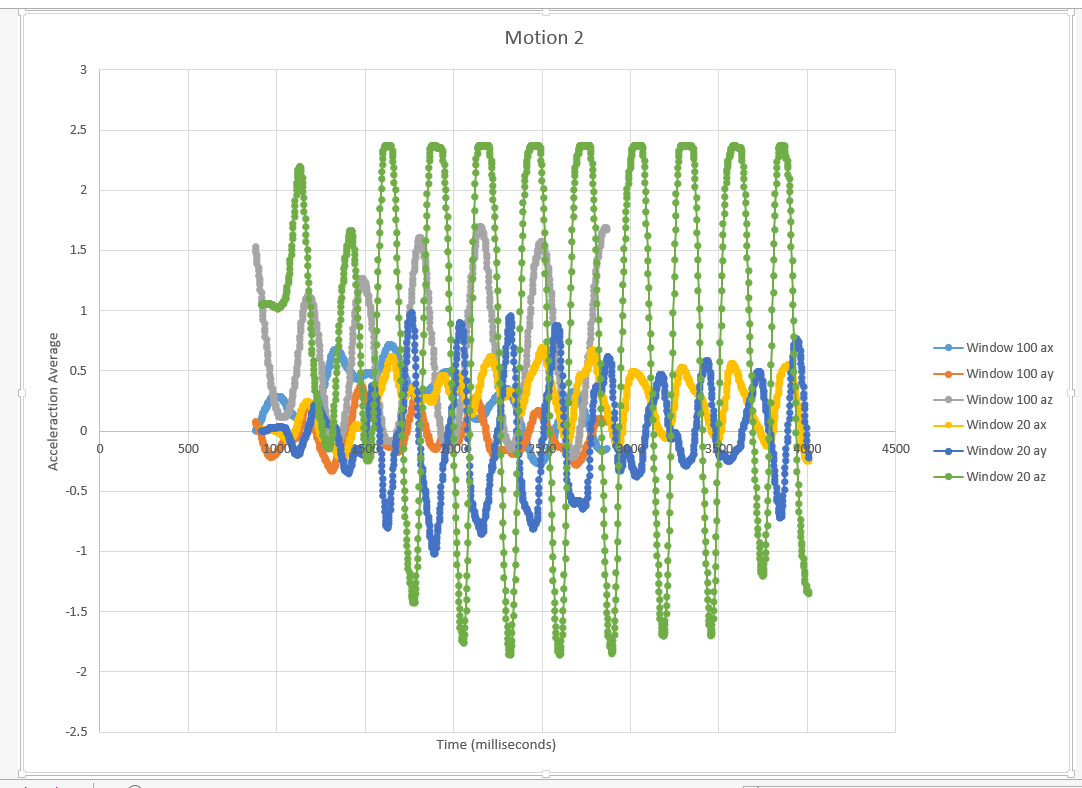
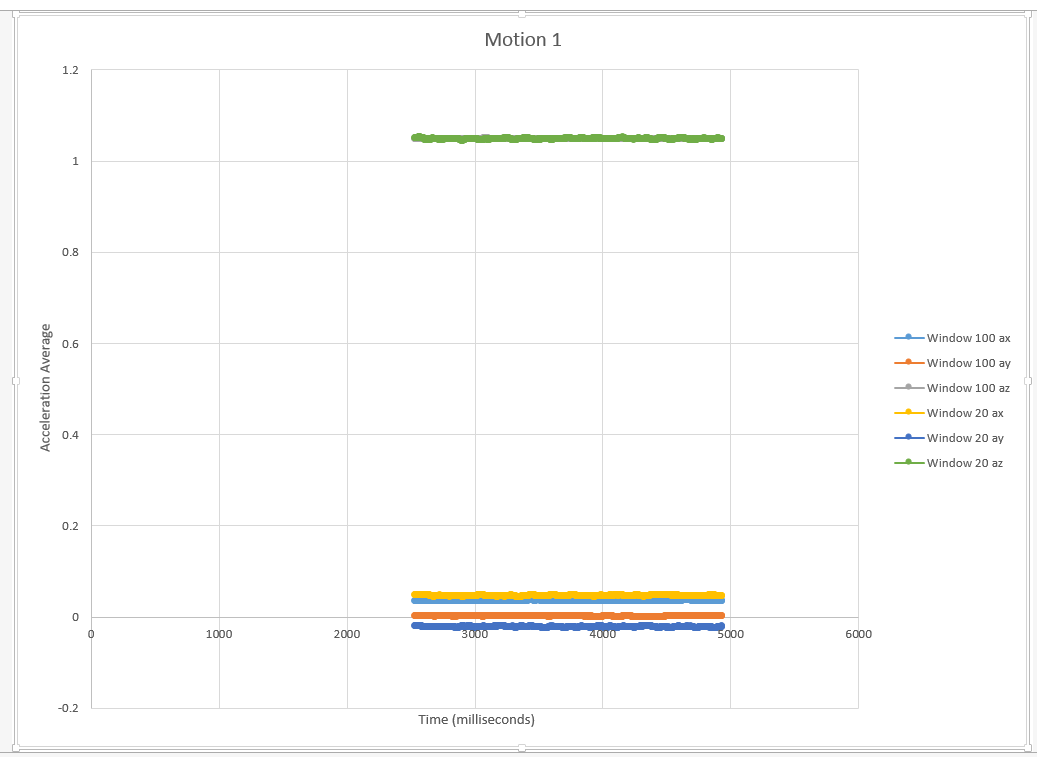
**Comments**

(Motion 3 was a constant up and down motion)

1. As window length increased, it seems as though the graphs become smoother, and the max’s are greater and the min’s are lesser. I think this is because more values are taken into account when the window length is greater, and because more values are taken into account, the average is more accurate.
2. Motion 2’s graph data is very choppy, up/down, and close together. Motion 3’s graph data is smooth, gently rolls, and spread out/easy to read. I would be more inclined to use a long window length for each of these because it creates a more accurate average which in turn makes for a smoother graph.

****

****

****