Testing Environment:

Testing was carried out on a Windows machine, since our module is compatible with windows at the moment. We used Microsoft excel as a tool to manually compute data. Our module is a .exe file which is executable on the command line as well as doubling clicking the application. Tests were performed using both methods.

The limitation was that tests were carried out for a small number of entries, since it would be very time consuming to do so, for larger chunks of data. We assume that if it works for a small input, it should then work for a large input size.

Overview Of Test Data:

Using 10 trade entries, we tested our module by varying the value of n and th. Basically, there were 3 main test cases for which were analyzed:

1. Setting parameters n = 3 and threshold= 0.0005
2. Changing n = 6 and keeping threshold = 0.0005
3. Changing threshold = 0.0008 and keeping n = 3

Then, we carried out testing using other teams’ modules. We used the same parameter and input file and generated the output file. This helped us compare our results with theirs. To sum it up, we tested our module by varying every parameter one by one, while keeping the others constant.

Testing Process:

First, we chose 10 trade entries from the original input file, and loaded it onto an excel spreadsheet. Then we manually entered the MSM strategy formulas one by one, to compute whether a buy or a sell signal was to be generated. The R(t) values were calculated in a extra column using an excel formula. Next the SMA values were calculated in a new column using the R(t) column and an excel formula. The SMA column was then used to calculate the TSV(t) value in another column also using an excel formula. Finally we manually compared the TSV(t) value to our threshold to compute whether to buy or sell. We then saved this information into an output file. We used this process for each of the input data, firstly for 10 inputs, parameters n = 3, threshold = 0.0005, and concurrently ran it using our module. We did this for all three test cases.

Once that was done, we had 3 output files generated by running the input on the module and 3 output files, which were manually computed. By running a diff on each corresponding pair of files, we concluded that the output generated by the module matched the manually computed results.

Then we used other teams’ modules to do the same, and compared the output. Now, we wanted to test for speed performance. We found that our module was taking longer than expected. ( please add on stuff about performance testing)