**Homework 3: Matrix Multiplication Report.**

Two randomly generated floating point matrices with the size of power of 2 were multiplied using the following three types of methods by code implementation:-

1) Standard Iterative

2) Standard Recursive

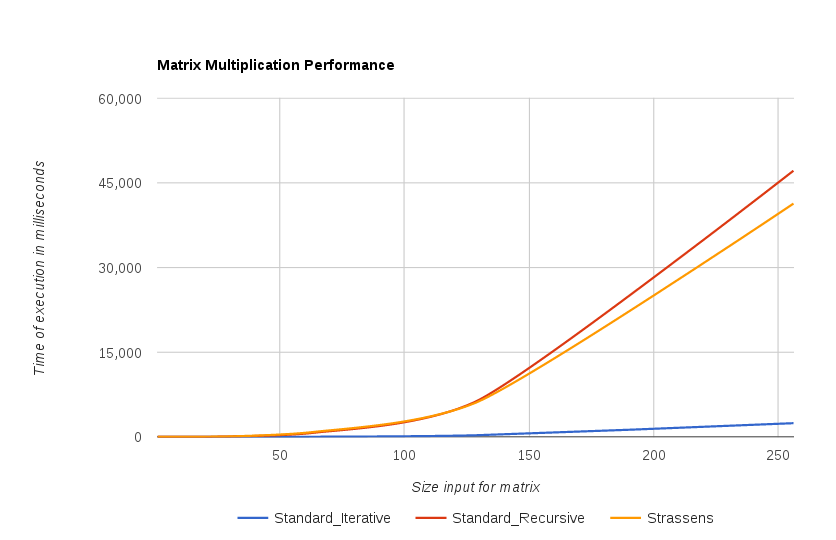
3) Strassens

The multiplication was repeated for sizes from 1, 2,...., 256 and the time required to compute the result for each of the above methods was noted.

Results:-

It is observed that Standard Iterative outperforms Standard Recursive and Strassens method significantly in terms of time of execution. However, amoungst the standard recursive and strassens, the former starts with a low cost compared to strassens but there is a crossover at around size range between 100 to 150 after which strassen is seen to marginally outperform standard recursive which is close to the required behavior, though this behavior is not consistent. Many a times strassen is seen not outperforming any of the other two methods and having the highest execution time.

Following is the graph plotted with the timing data collected when the required behavior was seen:-



**Strassen’s Method has the following practical implications:-**  
1) The submatrices in recursion take large memory and adds to processing time which is possibly the reason for it reporting higher execution time compared to standard iterative. This can be improved by performing in place operations for matrices, instead of creating copies.  
2) Limited precision of computer arithmetic on noninteger values, results in scope for large errors compared to standard method. This can be possibly improved by increasing the point precision of the non integer values to the max extent possible, however it trades off with the computational speed.

**Could Strassen's be improved for smaller matrices?**

I think that strassens multiplication implementation can be improved for smaller matrices by setting the recursion truncation point to a size such that matrices less than or equal to it (smaller recursive subproblems) can be solved using the conventional method and for which adding the recursive overhead would be no longer advantageous.