**Progress Report on Strassen Multiplication:**

Thus far, we have written code in C and Matlab that implement the recursive Strassen multiplication algorithm we discussed in our last report. The algorithm has turned out to be very memory intensive, mostly due to its recursive nature. To work around this, we decided to stop the recursion process before the size got too small. After some experimentation, we settled on a minimum matrix size of 64x64.

We went ahead and utilized OpenMP to parallelize the all the loops. The Strassen algorithm calls for the creation of some sub-matrices, M1-M7. This meant that each round of recursion performs several matrix addition and subtraction operations, as well as 7 matrix multiplications. The full algorithm is shown below. It’s implemented in three stages.

---stage 1---------------------------------------------------------------------------------------------------------------------------

---stage 2---------------------------------------------------------------------------------------------------------------------------

---stage 3---------------------------------------------------------------------------------------------------------------------------

Preliminary tests have shown our algorithm to be faster than a regular OpenMP matrix multiplication (with 3 nested loops) for matrices of size greater than 256 i.e. 256x256. This is consistent with the material that we have read on this algorithm. See the article “*Impact of Mixed-Parallelism on Parallel Implementations of Strassen and Winograd Matrix Multiplication Algorithms*”, by Desprez et.al. for further information on this method.























\mathbf{C}_{1,1} = \mathbf{M}_{1} + 
\mathbf{M}_{4} - \mathbf{M}_{5} + \mathbf{M}_{7}

\mathbf{C}_{1,2} = \mathbf{M}_{3} + 
\mathbf{M}_{5}

\mathbf{C}_{2,1} = \mathbf{M}_{2} + 
\mathbf{M}_{4}

\mathbf{C}_{2,2} = \mathbf{M}_{1} - 
\mathbf{M}_{2} + \mathbf{M}_{3} + \mathbf{M}_{6}