**COMPARISON**

**OF**

**DIFFERENT**

**MATRIX MULTIPLICATION**

**ALGORITHMS**

**Done by:**

**Apeksha Saxena – asaxena2**

**Anish Agarwal – anish**

**Sai Prajnan Emani – semani2**

Table of Contents

1. **Introduction**3
2. **Algorithms being analyzed**4 **2.1 Standard matrix multiplication**....................................................................................4
   1. **Strassen’s matrix multiplication**………………………………………………………………………………6
   2. **Multithreaded matrix multiplication**…………………………………………………………………….. 8
3. **Experiments and Results**…………………………………………………………………………………………………………….9

**3.1 Standard v/s Strassen’s**……………………………………………………………………………………………9

**3.2 Standard v/s Multi-threaded**…………………………………………………………………………………11

**3.3 Strassen’s v/s multi-threaded**……………………………………………………………………………….13

**3.4 Standard v/s Strassen’s v/s Multi-threaded**………………………………………………………….15

**4. References** ………………………………………………..…………………………………………………………….................17

**4.1 Tools** ……………………………………………………………………………..……………………………………...17

**5. Table of figures**……………………………………………………………………………………………………………..………...18

**1. INTRODUCTION**

The purpose of this project is to analyze and compare the common matrix multiplication algorithms. We compare the algorithms for square matrices of different orders, starting from a matrix of order 2 to a matrix of order 4096. Orders of matrices selected ,are powers of 2.

We compare the running times and the number of primitive operations for each of these algorithms under analysis.

An elaborate description for each of the algorithm used in the project has been explained in section 2 of the report.

Detailed textual, tabular and graphical reports of our findings have been documented.

All the experiments were run on a Lenovo ThinkPad T530 with the following specifications :

Operating System: Windows 8.1

RAM: 8GB

HDD: 500 GB

Processor: Intel Core i5-3320M CPU @ 2.60GHz

System type: x64-based processor

**2. Algorithms being analyzed**

We have selected the most common matrix multiplication algorithms for the purpose of our experiments, which are as follows:

1. Standard matrix multiplication

2. Strassen’s matrix multiplication

3. Multi-threaded matrix multiplication

**2.1 Standard Matrix Multiplication**

The standard matrix multiplication algorithm is the most common and naive algorithm for multiplying two matrices.

Let’s discuss the algorithm in a brief manner:

Consider multiplying two matrices A and B resulting in a resultant matrix C.

The necessary condition for every matrix multiplication algorithm is “Number of columns in matrix A = Number of rows in matrix B”.

The first step in the algorithm initializes every element in the resultant matrix C to zero.

Once the initialization is done, for each row in A and each column in B, multiply and sum the elements, and place the result in corresponding rows and columns of the matrix C.

2.1.1 Pseudo code

// Let A and B be square matrices of order n

for i = 1 to n

for j = 1 to n

C(i,j) = 0

for k = 1 to n

c(i,j) = c(i,j) + a(i,k) \* b(k,j)

end

end

end

**2.1.2 Running Time:**

The asymptotic running time of the standard matrix multiplication is O(n3), where n is the order of the matrices being multiplied.

The asymptotic run time implies that there would be a worst case of n3 number of primitive operations (scalar multiplications).

Following is a table consisting of results of the standard matrix multiplication algorithms for matrices of order (2, 4, 8… 4096) with the number of primitive operations and time (both in nanoseconds and seconds).

------------------------------------------------------------------------------------------

| Matrix Order | Number of primitive operations | Time (nanoseconds) | Time (seconds) |

| 2 | 8 | 790 | 0.00000079 |

| 4 | 64 | 3552 | 0.000003552 |

| 8 | 512 | 4750 | 0.00000475 |

| 16 | 4096 | 7105 | 0.000007105 |

| 32 | 32768 | 53684 | 0.000053684 |

| 64 | 262144 | 372635 | 0.000372635 |

| 128 | 2097152 | 3390429 | 0.003390429 |

| 256 | 16777216 | 28934105 | 0.028934105 |

| 512 | 134217728 | 308575981 | 0.308575981 |

| 1024 | 1073741824 | 14873009519 | 14.873 |

| 2048 | 8589934592 | 195720010753 | 195.72 |

| 4096 | 68719476736 | 1919886328074 | 1919.886 |

------------------------------------------------------------------------------------------

Table 1

From the above table it can be observed that,

The number of primitive operations required by the algorithm are n3 , where n is the order of the matrix.

**2.2 Strassen’s Matrix Multiplication**

The Strassen’s matrix multiplication algorithm was published by Volker Strassen in the year 1969. Strassen was the first person to point out that the standard matrix multiplication algorithm is not optimal.

Consider the multiplication of two square matrices of order 2.

Let the two matrices be A with elements A11,A12,A21,A22 where a11 indicates the element of row 1 column 1 , a12 indicates the element of row 1 column 2 and so on and B with elements B11,B12,B21,B22.

Using the standard matrix multiplication algorithm the number of scalar multiplications required for the multiplication of the matrices A and B would be 8 (n3).

Strassen determined an algorithm by which the number of multiplications is reduced to 7.

**2.2.1 Pseudo code**

The algorithm proceeds by defining a new set of matrices Mk where:

M1 = (A11 + A22)(B11 + B22)

M2 = (A21 + A22)B11

M3 = A11(B12 - B22)

M4 = A22(B21 - B11)

M5 = (A11 + A12)B22

M6 = (A21 - A11)(B11 + B12)

M7 = (A22 - A12)(B21 + B22)

The above 7 matrices are formed using only 7 multiplication operations.

Using the above matrices , we can determine the resultant matrix C :

C11 = M1 + M4 - M5 + M7

C12 = M3 + M5

C21 = M2 + M4

C22 = M1 - M2 + M3 + M6

**2.2.2 Running Time:**

The asymptotic running time of the Strassen’s matrix multiplication is O(n2.8084), where n is the order of the matrices being multiplied.

Following is a table consisting of results of the standard matrix multiplication algorithms for matrices of order (2, 4, 8… 4096) with the number of primitive operations and time (both in nanoseconds and seconds).

------------------------------------------------------------------------------------------

| Matrix Order | Number of primitive operations | Time (nanoseconds) | Time (seconds) |

| 2 | 7 | 790 | 0.00000079 |

| 4 | 49 | 3551 | 0.000003551 |

| 8 | 343 | 22495 | 0.000022495 |

| 16 | 2401 | 33157 | 0.000033157 |

| 32 | 16807 | 216709 | 0.000216709 |

| 64 | 117649 | 1542643 | 0.001542643 |

| 128 | 823543 | 10767702 | 0.010767702 |

| 256 | 5764801 | 77219920 | 0.07721992 |

| 512 | 40353607 | 666938315 | 0.666938315 |

| 1024 | 282475249 | 4555035943 | 4.555035943 |

| 2048 | 1977326743 | 31325069456 | 31.325069456 |

| 4096 | 12841287201 | 197220996778 | 197.220996778 |

Table 2

From the above table it can be observed that,

The number of primitive operations required by the algorithm is of the order O(n2.8074) and there is a contrasting difference when compared to the standard matrix multiplication algorithm as the matrix order increases.

**2.3 Multi-threaded Matrix Multiplication**

We have implemented the multi-threaded matrix multiplication in Java by inheriting the thread class.

Each thread is assigned the scalar multiplications pertaining to a single row of the resultant matrix.

Following is a table consisting of results of the standard matrix multiplication algorithms for matrices of order (2, 4, 8… 4096) with the number of primitive operations and time (both in nanoseconds and seconds).

------------------------------------------------------------------------------------------

| Matrix Order | Number of primitive operations | Time (nanoseconds) | Time (seconds) |

| 2 | 8 | 251055 | 0.000251055 |

| 4 | 64 | 400662 | 0.000400662 |

| 8 | 512 | 1316460 | 0.00131646 |

| 16 | 4096 | 5186105 | 0.005186105 |

| 32 | 32768 | 10786687 | 0.010786687 |

| 64 | 262144 | 21201923 | 0.021201923 |

| 128 | 2097152 | 30400173 | 0.030400173 |

| 256 | 16777216 | 44488076 | 0.044488076 |

| 512 | 134217728 | 144973291 | 0.144973291 |

| 1024 | 1073741824 | 22562177 | 0.22562177 |

| 2048 | 8589934592 | 2290874224 | 2.290874224 |

| 4096 | 68719476736 | 58440514142 | 58.440514142 |

Table 3

From the above table it can be observed that,

The number of primitive operations required by the algorithm is of the order O(n3) . The number of primitive operations required by this implementation and the standard matrix multiplication algorithm turns out to be same as there is no difference in the number of computations but the difference only lies in the way the computations are handled.

**3. Experiments and Results**

We have run a series of experiments on different sizes of matrices using the above discussed algorithms.

**3.1 Standard v/s Strassen’s Matrix Multiplication algorithms**

The standard and the Strassen's matrix multiplication algorithms were run for matrices of orders starting from 2 to a matrix of order 4096.

The following graph illustrates the results of the experiment:

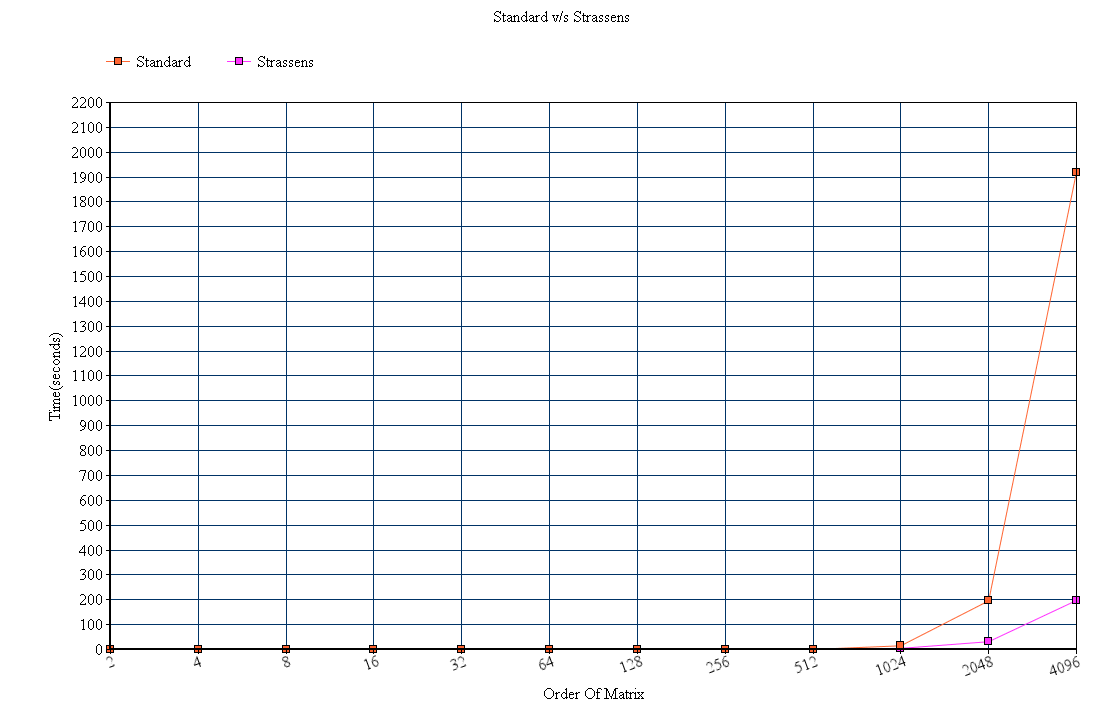


Figure 1 Standard v/s Strassen's matrix multiplication

From the above graph the following can be observed:

1. The running time of both the algorithms doesn’t show significant variance until the order of the matrices gets close to about a 1000.
2. At matrix size of 1024, a small variation of running times can be observed.

Standard matrix multiplication - 14.87 seconds.

Strassen's matrix multiplication - 4.550 seconds

1. At matrix size of 2048 a significant difference in running times can be seen :

Standard: 195.7200 seconds

Strassen’s: 31.325 seconds

1. At matrix size of 4096 a major difference is observed between the running time of both the algorithms :

Standard : 1919.886

Strassen’s: 197.220

The above observations indicate that the Strassen’s matrix multiplication algorithm is faster than the standard matrix multiplication algorithm for matrices of orders 1000 or more. Below that size the difference is negligible.

**3.2 Standard v/s Multi-threaded matrix multiplication**

The standard and the multi-threaded matrix multiplication algorithms were run for matrices of orders starting from 2 to a matrix of order 4096.

The following graph illustrates the results of the experiment:

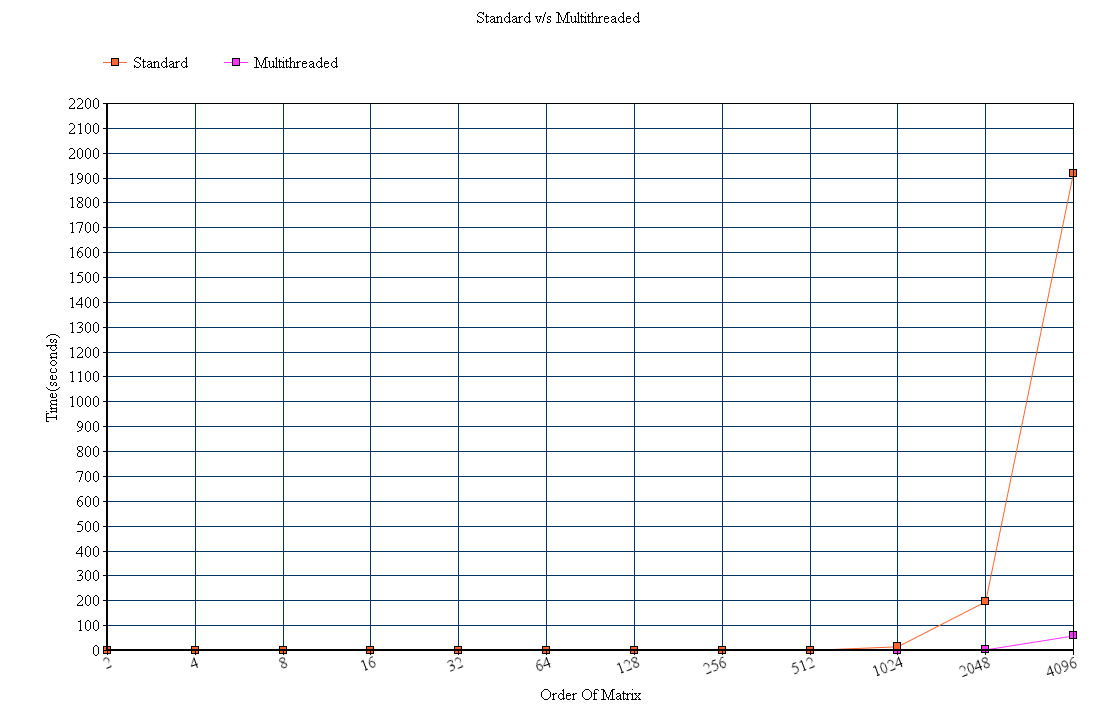


Figure 2 Standard v/s Multi-threaded matrix multiplication

From the above graph the following can be observed:

1. The multi-threaded matrix multiplication implementation has a longer running time compared to the standard matrix multiplication algorithm for matrices upto an order of about 250.
2. From the experiment it could be observed that the running times almost tend to be equal for matrices of order 256.
3. The running time of the multi-threaded matrix multiplication implementation starts getting better compared to the standard matrix multiplication algorithm for matrices of order 500 and over.

For matrix size of 512:

Standard matrix multiplication: 0.308575 seconds

Multi-threaded matrix multiplication: 0.144 seconds

1. The difference of running times starts to get significant with the increase in sizes of matrices being multiplied.
   1. Matrix size 1024 :
      * Standard matrix Multiplication: 14.87 seconds
      * Multi-threaded matrix multiplication: 0.2256 seconds
   2. Matrix size 2048 :
      * Standard matrix multiplication: 195.72 seconds
      * Multi-threaded matrix multiplication : 2.29 seconds
   3. Matrix size 4096 :
      * Standard matrix multiplication: 1919.886 seconds
      * Multi-threaded matrix multiplication:58.44 seconds

The above observations indicate that the multi-threaded matrix multiplication tends be more expensive and time consuming for matrix sizes less than or equal to 250, and standard matrix multiplication algorithm proves to be more time efficient for such matrices.

When the orders of the matrices being multiplied are 1000 and above the multi-threaded matrix multiplication is extremely time efficient and, a huge difference can be seen in the running times of both the algorithms.

**3.3 Strassen’s v/s Multi-threaded matrix multiplication**

The Strassen’s and the multi-threaded matrix multiplication algorithms were run for matrices of orders starting from 2 to a matrix of order 4096.

The following graph illustrates the results of the experiment:

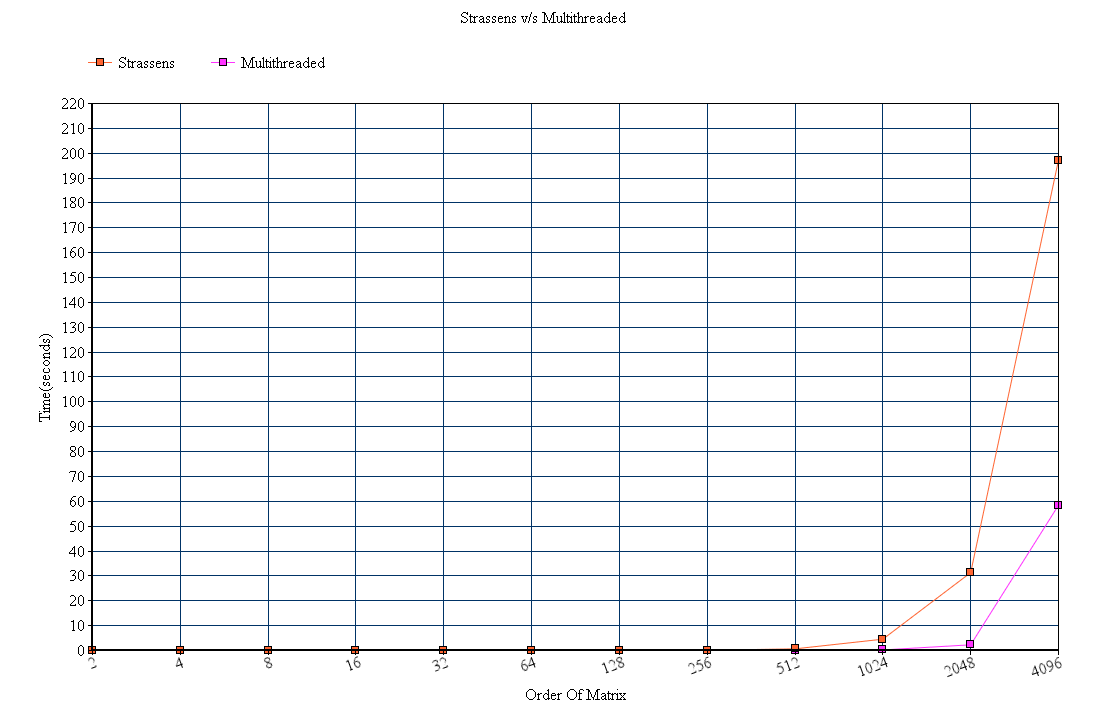


Figure 3 Strassen's v/s Multi-threaded matrix multiplication

From the above graph the following can be observed:

1. Multi-threaded matrix multiplication implementation is more time consuming than the Strassen’s algorithm for matrix sizes up to 200.
2. For matrix size of 256 :
   1. Strassen’s algorithm : 0.07 seconds
   2. Multi-threaded : 0.04 seconds

Which indicates that the multi-threaded algorithm is starting to get better than the Strassen’s algorithm comparing the running times.

1. For matrix size of 512:
   1. Strassen’s algorithm : 0.666 seconds
   2. Multi-threaded : 0.14 seconds
2. However, the difference in running times doesn’t get significant until the matrices are of size 1000 or above
3. Matrix size 1024 :

* Strassen’s matrix Multiplication: 4.55 seconds
* Multi-threaded matrix multiplication: 0.2256 seconds

1. Matrix size 2048 :

* Strassen’s matrix multiplication: 31.325 seconds
* Multi-threaded matrix multiplication : 2.29 seconds

1. Matrix size 4096 :

* Strassen’s matrix multiplication: 197.22 seconds
* Multi-threaded matrix multiplication:58.44 seconds

The above observations indicate that the multi-threaded matrix multiplication tends be more expensive and time consuming for matrix sizes less than or equal to 200, and the Strassen’s matrix multiplication algorithm proves to be more time efficient for such matrices.

When the orders of the matrices being multiplied are 1000 and above the multi-threaded matrix multiplication is extremely time efficient and, a significant difference can be seen in the running time.

**3.4 Standard v/s Strassen’s v/s Multi-threaded matrix multiplication**

The following graph illustrates the results of the experiment:

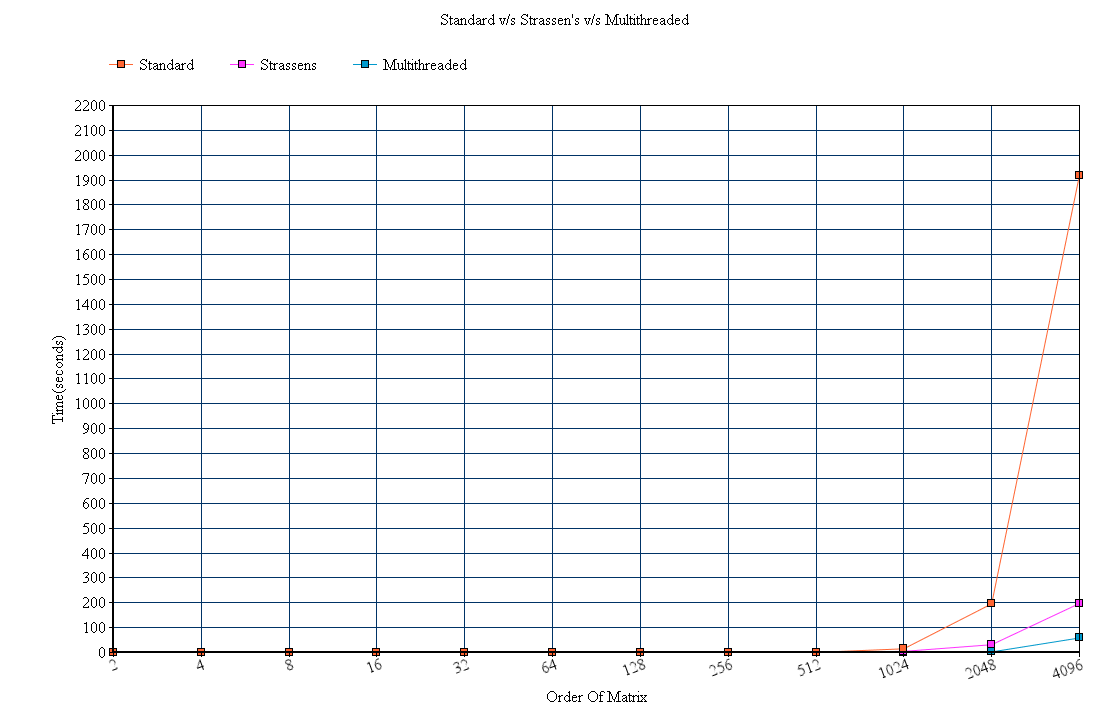


Figure 4 Standard v/s Strassen's v/s Multi-threaded matrix multiplication

From the above graph the following observations can be made:

1. The three algorithms almost have similar run times for matrices of sizes 500 or less. The multi-threaded implementation generally tends to be more time consuming and expensive when compared to standard and Strassen’s algorithm for these matrix sizes.
2. When the orders of the matrices are about 1000 or above the multi-threaded implementation is significantly better in terms of run time ,with Strassen’s algorithm in second place.

Matrix size 1024:

* Multi-threaded matrix multiplication : 0.22 seconds
* Strassen’s matrix multiplication : 4.55 seconds
* Standard matrix multiplication : 14.87 seconds

Matrix size 2048:

* Multi-threaded matrix multiplication : 2.29 seconds
* Strassen’s matrix multiplication : 31.325 seconds
* Standard matrix multiplication : 195.72 seconds

Matrix size 4096:

* Multi-threaded matrix multiplication : 58.440 seconds
* Strassen’s matrix multiplication : 197.22 seconds
* Standard matrix multiplication : 1919.88 seconds

1. The above findings indicate that for higher (>=1000) order matrices multi-threaded matrix multiplication is comparatively the fastest.

**4. References and Tools**

1. Standard Matrix Multiplication : <http://en.wikipedia.org/wiki/Matrix_multiplication>
2. Strassen’s Matrix Multiplication :
   1. <http://en.wikipedia.org/wiki/Strassen_algorithm>
   2. <https://github.com/MartinThoma/matrix-multiplication/tree/master/Java>
3. Multi-threaded Matrix multiplication
   1. <https://code.google.com/p/multithreading-matrix-multiplication-in-java/>
   2. <http://software.intel.com/en-us/articles/a-tale-of-two-algorithms-multithreading-matrix-multiplication>
   3. Introduction to Multithreaded programming : <http://www.tutorialspoint.com/java/java_multithreading.htm>

**4.1 Tools Used**

The following web tool has been used for generation of the required graphs:

Online Chart Tool: [http://www.onlinecharttool.com](http://www.onlinecharttool.com/)

**5. Table of Figures**

[Figure 1 Standard v/s Strassen's matrix multiplication 9](#_Toc373004599)

[Figure 2 Standard v/s Multi-threaded matrix multiplication 11](#_Toc373004600)

[Figure 3 Strassen's v/s Multi-threaded matrix multiplication 13](#_Toc373004601)

[Figure 4 Standard v/s Strassen's v/s Multi-threaded matrix multiplication 15](#_Toc373004602)