Strassen's algorithm

Strassen's algorithm is a famous algorithm used to speedup matrix products. It was originally presented to work on square matrices, on which it can reduce the complexity of matrix products from $\Theta(n^3)$ to $\Theta(n^{log_2(7)})$, where n is the size of the matrix.

However, it can be extended to work also on general matrix products of the form $A_{n\times m} \times B_{m\times p} = C_{n\times p}$.

To start, one can notice that the proof of correctness of the original Strassen's algorithm still holds for rectangular matrices whose sizes are powers of 2. In fact, if A, B and C are divided in blocks,

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},\ B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},\ C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

it is still true that:

$$\begin{aligned} C_{11} &= A_{11} \times B_{11} + A_{12} \times B_{21} \\ C_{12} &= A_{11} \times B_{12} + A_{12} \times B_{22} \\ C_{21} &= A_{21} \times B_{11} + A_{22} \times B_{21} \\ C_{11} &= A_{21} \times B_{12} + A_{22} \times B_{22} \end{aligned}$$

The only difference being that the submatrices are now rectangular. From this point onwards Strassen's proof can be applied also in this case, since all algebraic equations are still valid for rectangular matrices.

Some problems arise when one (or both) of the matrices A and B cannot be split exactly in 4 blocks, which happens immediately when at least one of the sizes n, m, p is odd or at some later recursive step when all sizes are even but at least one is not a power of 2 (i.e. it is not indefinitely divisible by 2).

To address this problem, I decided to zero-pad odd-sized matrices to the smallest even-sized matrices containing the original ones. To be clearer, before running a recursive step of Strassen's algorithm, a matrix $M_{n\times m}$ is zero-padded to $M'_{n'\times m'}$, with $n' = 2 \cdot \lceil \frac{n}{2} \rceil$ and $m' = 2 \cdot \lceil \frac{m}{2} \rceil$. This means that a matrix with an even number both of rows and columns undergoes no padding at all, while a matrix with an odd number both of rows and columns is provided with an additional bottom row of zeros and a right column of zeros.

Because of the properties of matrix products, this operation does not change the result of the single recursive step of Strassen's algorithm (if the result matrix C is reduced to its first n rows and p columns) and it is repeated every time an odd size is encountered.

About complexity, let's analyze the case of a square matrix multiplication with possibly odd size n. In this case, both matrices A and B are zero-padded to size $2 \cdot \lceil \frac{n}{2} \rceil$. Then, the complexity equation becomes:

$$T(n) = 7 \cdot T(\lceil \frac{n}{2} \rceil) + \Theta(n^2)$$

Which is exactly the complexity equation of Strassen's algorithm with the exception of a ceiling operation. This is because the additional cost of this modified algorithm lies in a fixed number of matrix sums and copies, which have cost $\Theta(n^2)$ that can be absorbed in the already present $\Theta(n^2)$ term.

To solve this recurrence, a recursion tree can be used. In this tree, choosing cn^2 as a representative for $\Theta(n^2)$, each node at level i has cost $c \cdot size_i^2$. The $size$ is n at level 0, $\lceil \frac{n}{2} \rceil$ at level 1, $\lceil \frac{\lceil \frac{n}{2} \rceil}{2} \rceil$ at level 2 and so on. These quantities can be upper bounded by $\frac{n+1}{2}$ at level 1, $\frac{n+3}{4}$ at level 2 and, in general, at level i, by $\frac{n+(2^i-1)}{2^i}$.

Since there are 7^i nodes at level i and the number of levels is at most $log_2(2n)$, the following equation holds:

$$T(n) \leq \sum_{i=0}^{log_2(2n)} 7^i c(\frac{n+(2^i-1)}{2^i})^2 = \sum_{i=0}^{log_2(2n)} (\frac{7}{4})^i c(n^2 + 2n(2^i-1) + (2^i-1)^2)^2 =$$
$$= cn^2 \sum_{i=0}^{log_2(2n)} (\frac{7}{4})^i + 2cn \sum_{i=0}^{log_2(2n)} (\frac{7}{4})^i (2^i-1) + c \sum_{i=0}^{log_2(2n)} (\frac{7}{4})^i ((2^i-1)^2)^2$$

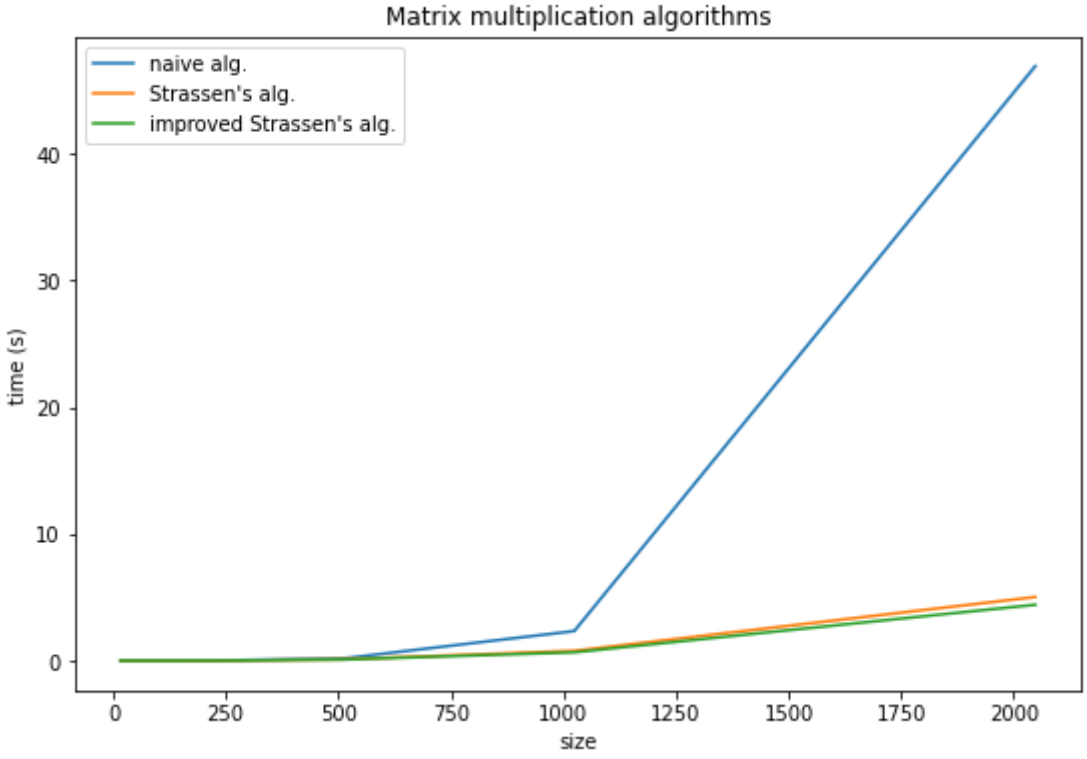
Analyzing only the higher order term:

$$cn^2 \sum_{i=0}^{log_2(2n)} (\frac{7}{4})^i = cn^2 \frac{(\frac{7}{4})^{log_2(2n)+1} - 1}{\frac{3}{4}} = \frac{4}{3}cn^2(\frac{7}{4})^{log_2(2n)+1} - \frac{4}{3}cn^2 = \frac{7}{3}cn^2(2n)^{log_2(\frac{7}{4})} - \frac{4}{3}cn^2 =$$
$$= \frac{49}{12}cn^2 n^{log_2(\frac{7}{4})} - \frac{4}{3}cn^2 = \frac{49}{12}cn^{log_2(7)} - \frac{4}{3}cn^2 \in O(n^{log_2(7)})$$

Hence, $T(n) \in O(n^{log_2(7)})$.

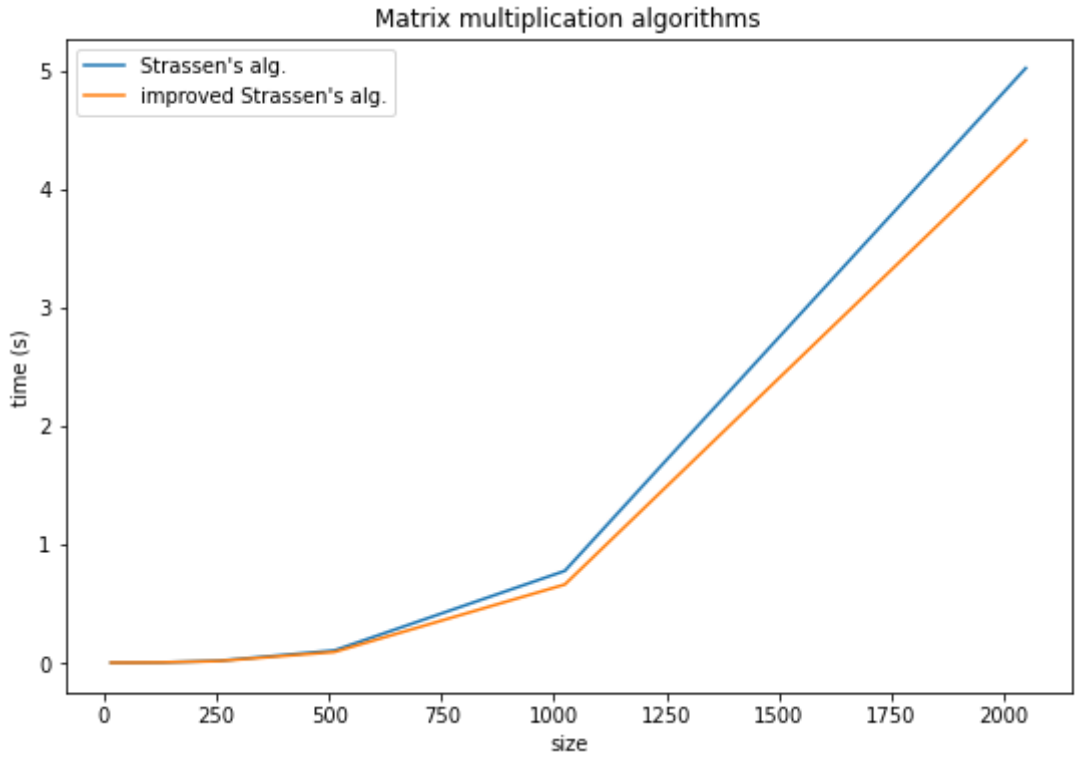
The source file `strassen.c` contains also another version of this algorithm, which improves performances by reducing the number of matrix allocations that have to be performed at each step.

This second version makes use of just two S matrices (they are 10 in usual Strassen's algorithm) and one P matrix (usually 7). The reduced number of matrices at disposal makes the algorithm just slightly more complex and requires some resets to zero of S matrices. However, it produces a small but significant improvement in performance as is visible from the following plots.



Both versions of Strassen's algorithm are already faster than the naive algorithm even for relatively small matrix sizes, but the difference becomes extremely significant for big matrices. One thing to point out here is that the algorithms were tried on rectangular matrices of sizes $(n-1 \times n, n \times n+1)$ and the $size$ reported on the x-axis of the graph is n.

To have a clearer idea of what happens to Strassen's algorithm and its improved version, the following plot reports only their two curves.



The improved algorithm achieves a visible improvement in time performance, yielding an execution time which is approximately 10% smaller than the one of Strassen's algorithm.