The Strassen Algorithm

CS 3C Final Project

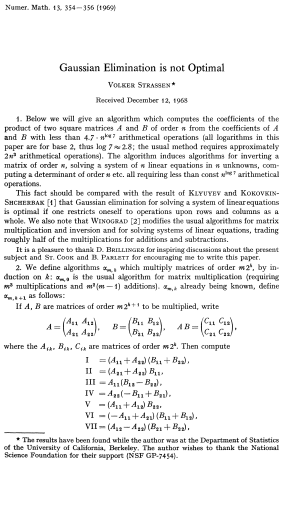
Sampson Mao

Foothill College

December 10, 2025

Introduction

The Strassen Algorithm is a matrix multiplication algorithm. Its namesake is from the person that discovered this algorithm, Volker Strassen. At the time, it was a significant discovery. It proved that the $\mathcal{O}(n^3)$ time complexity of matrix multiplication was not optimal.



Numer. Math. 13, 354–356 (1969)

Gaussian Elimination is not Optimal

Volker Strassen*

Received December 12, 1968

1. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with less than $4.7 \cdot n^{\log 7}$ arithmetical operations (all logarithms in this paper are for base 2, thus $\log 7 \approx 2.8$; the usual method requires approximately $2n^3$ arithmetical operations). The algorithm induces algorithms for inverting a matrix of order n, solving a system of n linear equations in n unknowns, computing a determinant of order n etc. all requiring less than const $\cdot n^{\log 7}$ arithmetical operations.

This fact should be compared with the result of Klyuyev and Kokovkin-Shcherbak [1] that Gaussian elimination for solving a system of linear equations is optimal if one restricts oneself to operations upon rows and columns as a whole. We also note that Winograd [2] modifies the usual algorithms for matrix multiplication and inversion and for solving systems of linear equations, trading roughly half of the multiplications for additions and subtractions.

It is a pleasure to thank D. Brillinger for inspiring discussions about the present subject and Sr. Cook and B. Parlett for encouraging me to write this paper.

2. We define algorithms $a_{m,k}$ which multiply matrices of order $m2^k$, by induction on k: $a_{m,0}$ is the usual algorithm for matrix multiplication (requiring m^3 multiplications and $m^3(m-1)$ additions). $a_{m,k}$ already being known, define $a_{m,k+1}$ as follows:

If A, B are matrices of order $m2^{k+1}$ to be multiplied, write

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

where the A_{ik}, B_{ik}, C_{ik} are matrices of order $m2^k$. Then compute

$$\begin{aligned} \text{I} &= (A_{11} + A_{22})(B_{11} + B_{22}), \\ \text{II} &= (A_{21} + A_{22})\,B_{11}, \\ \text{III} &= A_{11}(B_{12} - B_{22}), \\ \text{IV} &= A_{22}(-B_{11} + B_{21}), \\ \text{V} &= (A_{11} + A_{12})\,B_{22}, \\ \text{VI} &= (-A_{11} + A_{21})(B_{11} + B_{12}), \\ \text{VII} &= (A_{12} - A_{22})(B_{21} + B_{22}), \end{aligned}$$

* The results have been found while the author was at the Department of Statistics of the University of California, Berkeley. The author wishes to thank the National Science Foundation for their support (NSF GP-7454).

Introduction

- Strassen was associate professor at the Department of Statistics at UC Berkeley when he made this discovery.
- In 1968, he became director of the Seminar for Applied Mathematics at the University of Zurich.
- During this time, he sent his findings to the mathematics journal Numerische Mathematik. His paper was published in 1969.
- He worked at the University of Zurich for 20 more years until he moved onto the University of Konstanz.
- 10 years later, he retired. He is 89 as of this presentation.

Background - Matrix Multiplication Divide and Conquer

In order to understand Strassen's algorithm, we first need to look at another way of computing the product of 2 matrices.

- First, we divide the square matrices into 4 submatrices:

$$A = \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right), B = \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right)$$

- After partitioning, $A_{11}, A_{12}, A_{21}, A_{22}$ and $B_{11}, B_{12}, B_{21}, B_{22}$ of size $n/2 \times n/2$
- Then computing the product $C = AB$ is like finding the product of two 2×2 matrices:

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

Background - Matrix Multiplication Divide and Conquer

- Each block of C is a result of 2 matrix multiplications and 1 addition:

$$C = \left(\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right) = \left(\begin{array}{c|c} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ \hline A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{array}\right)$$

- resulting in a total of 8 multiplications and 4 additions.
- This is done recursively for each submatrix until the base case when the submatrix is a scalar.

Background - Matrix Multiplication Divide and Conquer

This approach still takes $\mathcal{O}(n^3)$ due to having 8 multiplications, which can be found by solving the recurrence relation:

$$T(n) = 8T(n/2) + \Theta(n^2) \tag{1}$$

- For each recursive call, the algorithm works on $1/4$ the size of the original matrix, but we need to do 8 times as many multiplication operations as the previous call. I.e. there are 8 subproblems that are $1/4$ of the size.
- The second term is the time complexity for the additions/subtractions. The additions take $\Theta(n^2)$ time, because we have to perform addition operations on n^2 entries.

Strassen's Algorithm

Having 8 multiplications leads to $\mathcal{O}(n^3)$ performance. Can we compute the product using a fewer number of multiplications?

Indeed, Strassen was able to reduce the number of multiplications from 8 to 7. Instead of calculating each entry C_{ij} in C, he first computed these intermediate matrices

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22})B_{11} \\
M_3 &= A_{11}(B_{12} - B_{22}) \\
M_4 &= A_{22}(B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12})B_{22} \\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22})
\end{aligned}
$$

Strassen's Algorithm

Then used them to compute the entries for matrix $C = \left(\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right)$

$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

We can see that while Strassen reduced the number of multiplications by 1, he added many more additions (and subtractions). In this algorithm, there are a total of 18 additions/subtractions. Is this a good tradeoff?

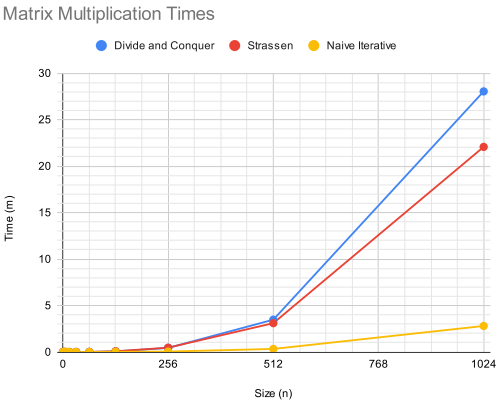
Strassen's Algorithm - Recurrance Relation

The recurrance relation for Strassen's Algorithm is

$$T(n) = 7T(n/2) + \Theta(n^2) \tag{2}$$

- In the first term, we decreased the number of subproblems by 1.
- The second term does not change, because we are still performing addition operations on multiples of n^2.
- Solving this, we get that this algorithm has a time complexity of $\mathcal{O}(n^{\log_2 7}) = \mathcal{O}(n^{2.807})$.

Experimental Results

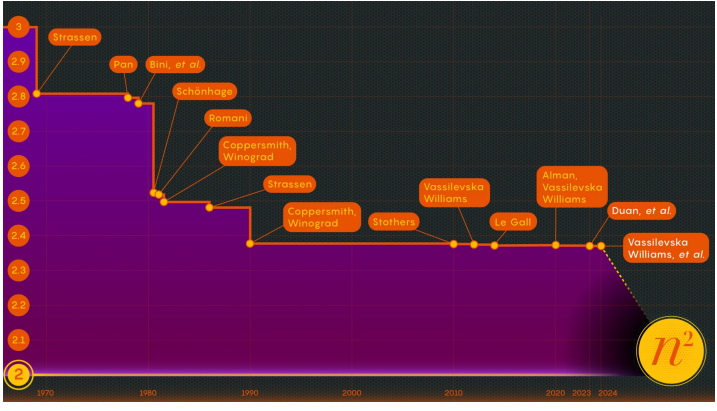


- Strassen does not do better than divide and conquer for smaller matrices, but we can see that it performs better after $n > 512$[†].
- Still, compared to the naive approach we did for week 3, these 2 algorithms seem much slower.
- Recursion in this case is slower due to the number of recursive calls. For each subtask, it is further split into 7-8 more.

[†]Used $n = 2^k$ to ensure submatrices were partitioned evenly. See appendix.

Can We Do Better?

Since Strassen's discovery, there have been attempts to find even faster algorithms. The current value for the exponent is 2.37155, which was achieved in 2024 by Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou.



Conclusion

While the big O of these algorithms is low, we can see that in practice they perform slowly. Strassen in particular is not stable and lead to rounding errors for larger inputs. Due to asymptotics, they only perform "fast" when the input is extremely large. Instead, libraries like Numpy are mostly written in C to decrease Python's overhead.

Numpy uses C/Fortran libraries like BLAS (Basic Linear Algebra Subprograms), which contain optimized algorithms for matrix operations. They contain optimizations such as

- Cache blocking - decreases the number of times memory needs to be accessed to fetch data.
- SIMD (Single instruction, multiple data) instructions - allows a single instruction to operate on multiple data points at once (vectorization).
- Multi-threading - parallelizing the calculations.

The End

Thank you!

References

[1] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, Aug. 1969, doi: https://doi.org/10.1007/bf02165411.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. Cambridge, Massachusetts: The Mit Press, 2022.

[3] S. Nadis, "New Breakthrough Brings Matrix Multiplication Closer to Ideal — Quanta Magazine," Quanta Magazine, Mar. 07, 2024. https://www.quantamagazine.org/new-breakthrough-brings-matrix-multiplication-closer-to-ideal-20240307/

[4] J. J. O'Connor and E. F. Robertson, "Volker Strassen - Biography," *Maths History*, 2025. https://mathshistory.st-andrews.ac.uk/Biographies/Strassen/ (accessed Dec. 03, 2025).

References

[5] R. Zadeh, "Lecture 3," Apr. 2016, Accessed: Dec. 03, 2025. [Online]. Available: https://stanford.edu/ rezab/classes/cme323/S16/notes/Lecture03/cme323_lec3.pdf

[6] A. Malik, "Optimizing Matrix Multiplication," Jul. 20, 2025. https://ashishmalik.in/post/faster_multiplication_of_matrix/ (accessed Dec. 03, 2025).

Appendix: Time Complexity Derivation for Strassen's Algorithm

$$
\begin{aligned}
T(n) &= 7T(n/2) + n^2 \\
&= 7(7T(n/4) + n^2/4) + n^2 \\
&= 7^2 T(n/4) + 7n^2/4 + n^2 \\
&= 7^2(7T(n/8) + n^2/16) + 7n^2/4 + n^2 \\
&= 7^3 T(n/8) + 7n^2/16 + 7n^2/4 + n^2 \\
&= \dots \\
&= 7^p T(n/2^p) + n^2 \sum_{k=1}^{p} \frac{1}{4^{k-1}} \\
&= 7^p T(1) + n^2 \sum_{k=0}^{p-1} \frac{1}{4^k}
\end{aligned}
$$

Appendix: Time Complexity Derivation for Strassen's Algorithm

$$
\begin{aligned}
T(n) &= 7^{\log n} T(1) + n^2 \sum_{k=0}^{\log n - 1} \frac{1}{4^k} \\
&= c7^{\log n} + n^2 \sum_{k=0}^{\log n - 1} \frac{1}{4^k} \\
&= cn^{\log 7} + n^2 \left(\frac{1 - \frac{1}{4^{\log n}}}{1 - \frac{1}{4}} \right) \\
&= cn^{\log 7} + \frac{4}{3} n^2 \left(1 - \frac{1}{n^2} \right)
\end{aligned}
$$

So the algorithm is $\mathcal{O}(n^{\log_2 7}) = \mathcal{O}(n^{2.807})$

Appendix: Rectangular Matrices and Matrices Of Size $n \neq 2^k$

- Both algorithms mentioned assume that the starting matrix can be subdivided evenly.
- To ensure that they work with matrices of all shapes and sizes, we can pad them with zeroes.
- One way is to pad both matrices to the next power of 2 in the beginning before computing the product. After getting the final result, we can drop the extra rows and columns.
- However, as written, the algorithms would involve potentially many computations with zeroes, which would slow them down.