Strassen's algorithm for matrix multiplication

Matrices are a fundamental concept in linear algebra. One of the most important operations on a matrix is matrix multiplication. To perform matrix multiplication on matrix A and matrix B, denoted as $C = A \times B$, A must have the shape of (m_1, n), and B (n, m_2), resulting in C (m_1, m_2), where $m_1, m_2, n \geq 1$, and $C_{ij} = \sum_k A_{ik} B_{kj}$, where $1 \leq i \leq m_1, 1 \leq j \leq m_2, 1 \leq k \leq n$. This means that it takes three nested loops to perform a matrix multiplication, which has the complexity of $O(m_1 * m_2 * n)$. The operation is of cubic order.

Strassen, in his 1969 paper, was the first to introduce a method to reduce the number of multiplications when performing matrix multiplication. The algorithm is a divide and conquer technique, where each step is the same and is done recursively at the smaller size of matrices. The initial requirement of Strassen's technique is that the matrices are square(shape $n \times n, n = 2m, m \geq 1$). In the traditional method, a multiplication of two $n \times n$ matrices requires n^3 and $(n-1)n^2$ component-wise multiplications and additions, respectively. For Strassen's matrix multiplication, the number of multiplications is $n^{log_2 7}$ and the number of additions is $6n^{log_2 7} - 6n^2$(Strassen, 1969; Murat Cenk & Hasan, 2017)(See appendix 2)

Strassen's algorithm works by breaking an $n \times n$ matrix into 4 $\frac{n}{2} \times \frac{n}{2}$ corner submatrices. Using his 7 equations(See appendix 1), the result matrix($n \times n$) can be calculated using the 4 corner matrices with only 7 multiplication operations rather than 8 in conventional method. The matrices are recursively divided by a factor of 2 at each step until reaching a theoretical size of 1×1, which takes the total of $log_2(n)$ steps. Each of the steps requires 7 multiplications, hence $7^{log_2(n)} = n^{log_2(7)}$. For matrices that are not originally square or have non-power-of-two dimensions, padding can be applied. The complexity order gets as low as $\omega = 2.371552$(Alman et al., 2024).

There have been many practical attempts to implement Strassen's matrix multiplication.. The purpose of the algorithm is to reduce the complexity of matrix multiplications, which have applications in various fields. Strassen's algorithm has been implemented on K-nearest neighbor(Rice, 2017). A tensor is a generalization of a matrix. Huang and others were the first to propose the idea of Tensor Contraction, which also employs Strassen to speed up the computation(Huang, 2018; Huang, Matthews, & van de Geijn, 2018). GPUs are designed for highly parallel workloads, including matrix and tensor operations. Despite the difference in nature, multiple attempts have been made to use Strassen's matrix multiplication on GPU,

with noticeable success(Ballard et al., 2012; Ballard et al., 2012; Zhao, 2021; Emmart & Weems, 2011). Basic Linear Algebra Subprograms (BLAS) is a standard set of building blocks for performing vector and matrix operations across multiple platforms(Netlib, n.d.). Based on BLAS, many researchers have implemented Strassen's algorithm on GPU like BLIS, CUTLASS(Huang, Yu, & Van de Geijn, 2020; Huang, Smith, Henry, & van de Geijn, 2016).

Although research has shown positive results, Strassen's algorithm remains impractical for real-world applications(Ballard et al., 2012). The underlying reason for this is the expensive overhead that the algorithm presents. In theory, given the number of operations for each case above, Strassen's algorithm will need $n > 654$(See appendix 3) to out-perform the conventional way. In experiments, n remains just as high; Conventional method is usually used once n gets low enough, and the lowest point is $n = 500$(Huang, Yu, & Van de Geijn, 2020). Other attempts have significantly higher switch-points at $n = 512$ (Sunday & Atabong, 2015), and $n \geq 1000$(D'Alberto & Nicolau, 2005; Benson & Ballard, 2015; Ballard et al., 2012). Ballard et al. (2012) claims to have proposed a method to reduce the overhead, still it is not until the matrix size is really high($n > 1000$) before a noticeable different can be seen.

Another issue with Strassen's technique is its lack of numerical stability. Numerical Stability is how algorithm introduces errors that affect the output results(Higham, 2020). In conventional method, matrix multiplication involves only multiplication and addition operations, whereas in Strassen's method, a number of subtraction operations is used on intermediate matrices($\frac{n}{2} \times \frac{n}{2}$ corner matrices). Subtraction introduces potential cancellation, where nearly equal values result in the loss of significant digits, and rounding errors accumulate in these intermediate matrices (Goldberg, 1991); This makes Strassen's method more prone to numerical instability(De Silva, Gustafson, & Wong, 2018, p. 682; Emmart & Weems, 2011). Researches have shown that Strassen's algorithm can be numerical stable, but they use norm-wise numerical stability, which is a weaker method than component-wise numerical stability(De Silva, Gustafson, & Wong, 2018; Demmel, Dumitriu, Holtz, & Kleinberg, 2007).

In conclusion, Strassen was the first to propose a method for reducing the computational cost of multiplying $2\ n \times n$ matrices. The method is a recursive divide and conquer algorithm that performs the matrix multiplication using 7 multiplication operations on $4\ \frac{n}{2} \times \frac{n}{2}$ corner matrices. Theoretically, the method reduces the complexity to $O(n^{log_2 7}) \approx O(n^{2.807})$. Strassen's matrix multiplication has seen multiple experimental implementations with positive results. However, due to several reasons including but not limited to having large overhead, and being numerically unstable, Strassen's matrix multiplication's practical application remains unclear.

References

1. Strassen, V. (1969). Gaussian elimination is not optimal. *Numerische Mathematik*, *13*(4), 354–356. https://doi.org/10.1007/BF02165411
2. STRASSEN, V. (1987). Relative bilinear complexity and matrix multiplication.. Journal für die reine und angewandte Mathematik, 1987(375-376), 406-443. https://doi.org/10.1515/crll.1987.375-376.406(update)
3. Ahmad, A., Du, L., & Zhang, W. (2024). Fast and practical Strassen's matrix multiplication using FPGAs. arXiv. https://arxiv.org/abs/2406.02088
4. Ballard, G., Demmel, J., Holtz, O., Lipshitz, B., & Schwartz, O. (2012). Communication-optimal parallel algorithm for Strassen's matrix multiplication. In Proceedings of the Twenty-Fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures (pp. 193–204). Association for Computing Machinery. https://doi.org/10.1145/2312005.2312044
5. Huang, J. (2018). Practical fast matrix multiplication algorithms (Doctoral dissertation). The University of Texas at Austin. Available at https://repositories.lib.utexas.edu/items/6d66e120-5164-4879-b2f6-f13fa9e706e6
6. Sunday, A., & Atabong, T. (2015). Determining the effects of cross-over point on the running time of Strassen matrix multiplication algorithm. *British Journal of Mathematics & Computer Science*, *10*, 1–12. https://doi.org/10.9734/BJMCS/2015/18772
7. Huang, J., Yu, C. D., & Van de Geijn, R. A. (2020). Strassen's algorithm reloaded on GPUs. *ACM Transactions on Mathematical Software*, *46*(1), Article 1. https://doi.org/10.1145/3372419
8. van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22–30. https://doi.org/10.1109/MCSE.2011.37
9. DeepMind. (2022, October 5). *Discovering novel algorithms with AlphaTensor*. DeepMind Blog. https://deepmind.google/discover/blog/discovering-novel-algorithms-with-alphatensor/
10. MAGMA (n.d.). *Matrix multiplication*. University of Sydney. Retrieved Feb 18 2025, from https://magma.maths.usyd.edu.au/magma/overview/2/16/8/

11. Huang, J., Smith, T. M., Henry, G. M., & van de Geijn, R. A. (2016). *Implementing Strassen's algorithm with BLIS* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.1605.01078

12. D'Alberto, P., & Nicolau, A. (2005). Adaptive Strassen and ATLAS's DGEMM: A fast square-matrix multiply for modern high-performance systems. *Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05)*, 8 pp.–52. https://doi.org/10.1109/HPCASIA.2005.18

13. Benson, A. R., & Ballard, G. (2015). A framework for practical parallel fast matrix multiplication. *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2015)*, 42–53. https://doi.org/10.1145/2688500.2688513

14. Ballard, G., Demmel, J., Holtz, O., Lipshitz, B., & Schwartz, O. (2012). *Communication-optimal parallel algorithm for Strassen's matrix multiplication* [Preprint]. arXiv. https://arxiv.org/abs/1202.3173

15. D'Alberto, P. (2023). *Strassen's matrix multiplication algorithm is still faster* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2312.12732

16. Dumitrescu, B. (1998). Improving and estimating the accuracy of Strassen's algorithm. *Numerische Mathematik, 79*(4), 485–499. https://doi.org/10.1007/s002110050348

17. Zhao, K. (2021). Strassen algorithm and its performance. *International Journal of Computational Engineering, 7*(4), 75–81. https://doi.org/10.6919/ICJE.202104_7(4).0013

18. De Silva, H., Gustafson, J. L., & Wong, W.-F. (2018). Making Strassen matrix multiplication safe. *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, 173–182. https://doi.org/10.1109/HiPC.2018.00028

19. Higham, N. J. (1992). Stability of a Method for Multiplying Complex Matrices with Three Real Matrix Multiplications. SIAM Journal on Matrix Analysis and Applications, 13(3), 681–687. doi:10.1137/0613043

20. Demmel, J., Dumitriu, I., Holtz, O., & Kleinberg, R. (2007). Fast matrix multiplication is stable. *Numerische Mathematik, 106*(2), 199–224. https://doi.org/10.1007/s00211-007-0061-6

21. Rice, L. (2017). Performance Optimization for the K-Nearest Neighbors Kernel using Strassen ' s Algorithm.

22. Nair, R. (2015). Evolution of memory architecture. *Proceedings of the IEEE, 103*(8), 1331–1345. https://doi.org/10.1109/JPROC.2015.2435018

23. Chang, K. K. (2017). *Understanding and improving the latency of DRAM-based memory systems*. arXiv. https://arxiv.org/abs/1712.08304

24. Huang, J., Matthews, D. A., & van de Geijn, R. A. (2018). Strassen's algorithm for tensor contraction. *SIAM Journal on Scientific Computing*, *40*(3), C305–C326. https://doi.org/10.1137/17M1135578

25. Emmart, N., & Weems, C. C. (2011). High precision integer multiplication with a GPU using Strassen's algorithm with multiple FFT sizes. *Parallel Processing Letters*, *21*(3), 359–375. https://doi.org/10.1142/S0129626411000266

26. Murat Cenk, M., & Hasan, M. A. (2017). On the arithmetic complexity of Strassen-like matrix multiplications. Journal of Symbolic Computation, 80(2), 484–501. https://doi.org/10.1016/j.jsc.2016.07.004

27. Alman, J., Duan, R., Williams, V. V., Xu, Y., Xu, Z., & Zhou, R. (2024). More asymmetry yields faster matrix multiplication. *arXiv preprint*, arXiv:2404.16349. https://doi.org/10.48550/arXiv.2404.16349

28. Netlib. (n.d.). *Basic Linear Algebra Subprograms (BLAS)*. Retrieved [Feb 19 2025], from https://www.netlib.org/blas/

29. Higham, N. J. (2020, August 4). *What is numerical stability?* Nick Higham's Blog. https://nhigham.com/2020/08/04/what-is-numerical-stability/

30. Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, *23*(1), 5–48. https://doi.org/10.1145/103162.103163

Appendix

1. Strassen's matrix multiplication

$$\text{Let } A \text{ and } B \text{ be } 2 \times 2 \text{ matrices:}$$
$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \qquad B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

Strassen defined 7 intermediate matrix multiplications.

$$
\begin{aligned}
P_1 &= (a+d)(e+h) \\
P_2 &= (c+d)e \\
P_3 &= a(f-h) \\
P_4 &= d(g-e) \\
P_5 &= (a+b)h \\
P_6 &= (c-a)(e+f) \\
P_7 &= (b-d)(g+h)
\end{aligned}
$$

The matrix $C = A \times B$

$$
C = \begin{bmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 + P_3 - P_2 + P_6 \end{bmatrix}
$$

2. Strassen’s algorithm, number of multiplication and addition operations.

a. For multiplication.

We define the recurrence:

$$M(n) = 7M(\frac{n}{2}) \quad (1)$$

Because it is recursive $M(\frac{n}{2}) = 7M(\frac{n}{4})$, substitute it into the (1), we have:

$$M(n) = 7 * 7 * M(\frac{n}{4})$$

The size reduction factor is 2, we will have $log_2(n)$ recursion before the size of the matrices gets to 1×1. Therefore:

$$M(n) = 7^{log_2(n)} = n^{log_2(7)}$$

b. For Addition/subtraction

There are a total of 18 addition/subtraction in Strassen’s formula, each operation is done on two $\frac{n}{2} \times \frac{n}{2}$ matrices, so we have $f(n) = 18(\frac{n}{2})^2$. We define the recurrence:

$$A(n) = 7A(\frac{n}{2}) + 18(\frac{n}{2})^2$$

Follow the same steps as in multiplication recurrence, we expand the $A(n)$ to get the summation:

$$A(n) = \sum_{0}^{log_2(n)} 18 * 7^i * (\frac{n}{2^{i+1}})^2 = 18n^2 \sum_{0}^{log_2(n)} \frac{7^i}{(2^{i+1})^2} = \frac{18}{4}n^2 \sum_{0}^{log_2(n)} (\frac{7}{4})^i$$

This is a geometric sum with $k = \frac{7}{4}, r = log_2(n)$, so we can rewrite it as:

$$A(n) = \frac{18}{4}n^2(\frac{1 - (\frac{7}{4})^{log_2(n)}}{1 - \frac{7}{4}}) = \frac{18}{4}n^2(\frac{1 - (\frac{7}{4})^{log_2(n)}}{-\frac{3}{4}}) = -6n^2(1 - \frac{7^{log_2(n)}}{4^{log_2(n)}})$$

We have $a^{log_b(c)} = c^{log_b(a)}$, so $4^{log_2(n)} = n^{log_2(4)} = n^2$, and $7^{log_2(n)} = n^{log_2(7)}$.

$$A(n) = -6n^2 \frac{n^2 - n^{log_2(7)}}{n^2} = 6n^{log_2(7)} - 6n^2$$

3. Strassen vs conventional total operations