Alternative Methods of Matrix Multiplication

Jack Ruder

May 2, 2021

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

Algorithm 1 pseudocode for $blockmult(A, B, bs)$

if $bs = 1$ **then**
  return $A \cdot B$
end if
for $i = 1$ to 2 **do**
  for $j = 1$ to 2 **do**
    $C[i,j] = blockmult(A[i,1], B[1,j], \frac{bs}{2}) +$
    $blockmult(A[i,2], B[2,j], \frac{bs}{2})$
  end for
end for

- 8 multiplications are needed.
- 4 additions are needed.
- Total arithmetic operations are $M(2n) = 8M(n) + 4n^2$.
- Or, $M(n) = 2n^3 - n^2$.
- This is the same complexity as entry-by-entry matrix multiplication.

Strassen's multiplication only requires 7 multiplications.

Strassen's algorithm for matrix multiplication

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

$$\begin{aligned}
C_{11} &= M_1 + M_4 - M_5 + M_7 & C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 & C_{22} &= M_1 + M_3 - M_2 + M_6.
\end{aligned}$$

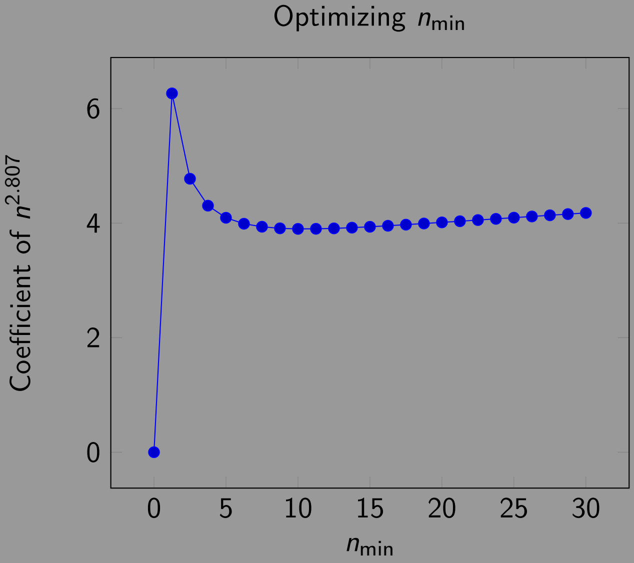
Let $A = \begin{bmatrix} -1 & -1 \\ 4 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} -3 & 1 \\ 2 & 1 \end{bmatrix}$. To find AB we compute

$$\begin{aligned} M_1 &= -2 \\ M_2 &= -18 \\ M_3 &= 0 \\ M_4 &= 10 \\ M_5 &= -2 \\ M_6 &= 10 \\ M_7 &= -9 \end{aligned} \qquad \begin{aligned} C_{11} &= 1 \\ C_{12} &= -2 \\ C_{21} &= -8 \\ C_{22} &= 6. \end{aligned}$$

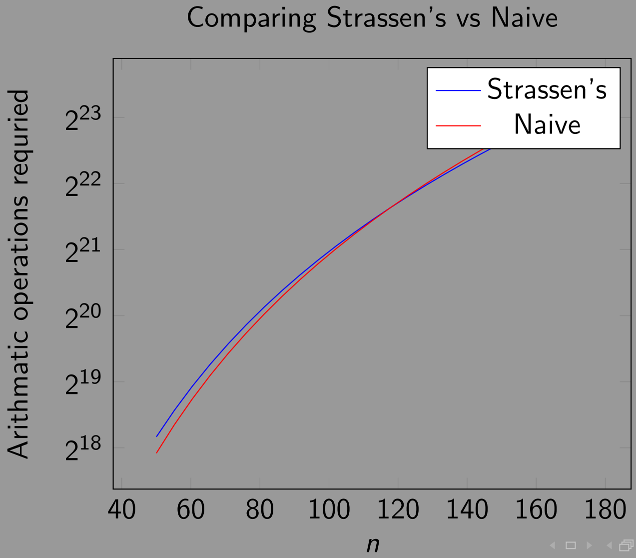
Note that since this was a 2×2 example, we don't need to recurse, and treat the 1×1 blocks as scalars.

- 7 multiplications are needed per recursion.
- 18 additions are needed per recursion.
- If the crossover point is n_{min}, then $\log_2 n - \log_2 n_{\text{min}}$ recursions are needed.
- Total arithmetic operations are $M(n) \approx n^{\log_2 7} \frac{M(n_{\text{min}})+6n_{\text{min}}^2}{n_{\text{min}}^{\log_2 7}}$
- Strassen's with naive complexity becomes $M(n) \approx n^{\log_2 7} \frac{2n_{\text{min}}^3+5n_{\text{min}}^2}{n_{\text{min}}^{\log_2 7}}$.

- We see a value of 8 for n_{min} is close to optimal.



- The complexity is then $M(n) \sim 5.00n^{2.807}$.
- Is this really practical?



- Best case scenario, Strassen's only yields improvements when $n \geq 128$
- In practice, n is much higher.
  - Matrices are best represented in column-major or row-major orders.
  - These storage schemes are more contiguous in memory than block storage schemes.
  - Strassen's is not cache-optimal as a result.

Triangular matrices can be multiplied faster than regular matrices because of the zeroes.

Block Triangular Multiplication

For upper triangular A and upper triangular B,

$$\begin{aligned} AB &= \begin{bmatrix} A_{11}B_{11} + A_{12}0 & A_{11}B_{12} + A_{12}B_{22} \\ 0B_{11} + A_{22}0 & 0B_{12} + A_{22}B_{22} \end{bmatrix} \\ &= \begin{bmatrix} A_{11}B_{11} & A_{11}B_{12} + A_{12}B_{22} \\ 0 & A_{22}B_{22} \end{bmatrix}, \end{aligned}$$

where A_{11}, B_{11}, A_{22} and B_{22} are upper triangular.

This approaches $O(n^2)$.

Strassen's in the triangular case is disappointing.

Strassen's Triangular Multiplication

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) & M_2 &= A_{22}B_{11} \\
M_3 &= A_{11}(B_{12} - B_{22}) & M_4 &= A_{22}(-B_{11}) \\
M_5 &= (A_{11} + A_{12})B_{22} & M_6 &= (-A_{11})(B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22})B_{22}. \\
M_8 &= M_1 - M_2 \\
C_{11} &= M_8 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= 0 \\
C_{22} &= M_8 + M_3 + M_6.
\end{aligned}
$$

- We still need 6 multiplications.
  - 2 normal, 4 triangular.
- 12 additions are needed.
  - 9 normal, 3 triangular.
- Less arithmetic operations are needed in the standard block triangular algorithm, in both additions and multiplications.
- Strassen's here can't compete.

For nonsingular A, the block LDU decomposition is

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ 0 & I \end{bmatrix}.$$

where $\Delta = A_{22} - A_{21}A_{11}^{-1}A_{12}$. The inverse comes easily,

$$\begin{aligned} A^{-1} &= \begin{bmatrix} I & -A_{11}^{-1}A_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{11}^{-1} & 0 \\ 0 & \Delta^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{21}A_{11}^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}\Delta^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}\Delta^{-1} \\ -\Delta^{-1}A_{21}A_{11}^{-1} & \Delta^{-1} \end{bmatrix}. \end{aligned}$$

Strassen's Inversion relies on these steps in order.

Strassen's algorithm for matrix inversion

1. $M_1 = A_{11}^{-1}$
2. $M_2 = A_{21} M_1$
3. $M_3 = M_1 A_{12}$
4. $M_4 = A_{21} M_3$
5. $M_5 = M_4 - A_{22}$
6. $M_6 = M_5^{-1}$
7. $(A^{-1})_{12} = M_3 M_6$
8. $(A^{-1})_{21} = M_6 M_2$
9. $M_7 = M_3 (A^{-1})_{21}$
10. $(A^{-1})_{11} = M_1 - M_7$
11. $(A^{-1})_{22} = -M_6$.

- The most expensive computation is matrix multiplication.
- We can find the inverse in $O(mul(n))$.

- Strassen's provides speedups in matrix multiplication for large matrices.
- The results of faster matrix multiplication are quicker inversions.
- Given the lack of versatility in other types of matrices, Strassen's is not a great all-around tool.