

# Jacob Minz's Blog

Monday, 4 May 2015

## Derivation of Strassen's Algorithm for the multiplication of $2 \times 2$ matrices.

### Abstract

Strassen's multiplication algorithm for the multiplication of  $2 \times 2$  matrices using seven multiplication instead of the naive 8 multiplication heralded a new era of research into asymptotically more efficient algorithms. It showed that the multiplication of two  $n \times n$  matrices could be achieved in less than  $O(n^3)$  time. In this blog, we attempt to present a derivation of this seminal work, which only requires a modest background in linear algebra.

### Introduction

The  $2 \times 2$  matrix multiplication  $A \cdot B = C$  is usually written as:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

The classic "definition" of matrix multiplication says that

$$C = \begin{bmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} & a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} & a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{bmatrix}$$

As can be seen an upper bound on the number of multiplications, is quickly identified to be 8.

In vector notation, we have row vectors for  $A$  and column vectors for  $B$ :

$$\begin{aligned} A_1 &= a_{11}e_1 + a_{12}e_2 \\ A_2 &= a_{21}e_1 + a_{22}e_2 \end{aligned}$$

and

$$\begin{aligned} B_1 &= b_{11}e_1 + b_{21}e_2 \\ B_2 &= b_{12}e_1 + b_{22}e_2 \end{aligned}$$

Then

$$C = \begin{bmatrix} A_1 \cdot B_1 & A_1 \cdot B_2 \\ A_2 \cdot B_1 & A_2 \cdot B_2 \end{bmatrix}$$

### Strassen's Algorithm

Strassen's algorithm provided the first lowering of the upper bound for the matrix multiplication of  $2 \times 2$  matrices. Replicating the construction from wikipedia [page](#) for easy reference:

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

### About Me

Unknown

[View my complete profile](#)

### Blog Archive

▼ 2015 (3)

► July (2)

▼ May (1)

[Derivation of Strassen's Algorithm for the multipl...](#)

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

### Main ideas behind the derivation

We now outline the main ideas behind the derivation.

1. The dot-product in vector notation remains invariant with *simultaneous indices rotation*.
2. Keeping the dot-product constant for  $c_{i,j}$ , we explore various transformations of the vectors  $A_i$  and  $B_j$ .
3. We do not restrict the vector transformation to 2-dimensions, we may transform 2-D vectors to 4-D, if the dot-product invariance is maintained.
4. When transforming to higher dimension vectors, we try to reuse as much computation from the same-dimension transforms. This is suspiciously similar to *embedding* in Group Theory.
5. We search for symmetries in the computation of the various dot-product.

WLOG, we transform  $c_{21} = A_2 \cdot B_1 = \hat{A}_2 \cdot \widehat{B}_1$ , and then use simultaneous indices rotation to get  $c_{12}$ . Similarly, we transform  $c_{11} = A_1 \cdot B_1 = \tilde{A}_1 \cdot \widetilde{B}_1$ , and then use simultaneous indices rotation to get  $c_{22}$ . We shall see that we identify one *symmetric computation* in the computation set.

### Transformations and Computations

We now propose the transformations and computations, which leads to Strassen's algorithm. For transformation (1), we will use the following transformer  $2 \times 2$  matrices, such that:

$$L_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad L_1^{-1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad L_1 \cdot L_1^{-1} = I$$

The matrix  $L_1$  is easily recognized as the first of *Pascal's matrices*. Here is the [link](#).

Transformation is as follows:

$$\begin{aligned}c_{21} &= \hat{A}_2 \cdot \widehat{B}_1 = [a_{21} \quad a_{22}] \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} \\c_{21} &= [(a_{21} + a_{22}) \quad a_{22}] \cdot [b_{11} \quad (-b_{11} + b_{21})]^T\end{aligned}$$

Calculating  $c_{21}$  and using simultaneous rotation of both indices for  $c_{21}$ , we arrive at:

$$\begin{aligned}c_{21} &= [(a_{21} + a_{22})b_{11} + a_{22}(-b_{11} + b_{21})] \\c_{12} &= [(a_{12} + a_{11})b_{22} + a_{11}(-b_{22} + b_{12})]\end{aligned}$$

For transformation (2), we will use the following transformer  $4 \times 4$  matrices, such that:

$$L_2 = \begin{bmatrix} L_1 & 0 \\ I & L_1^{-1} \end{bmatrix} \quad \text{and} \quad L_2^{-1} = \begin{bmatrix} L_1^{-1} & 0 \\ -I & L_1 \end{bmatrix} \quad \text{and} \quad L_2 \cdot L_2^{-1} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} = I_{4 \times 4}$$

A point to note here is that  $L_2$  is not a Pascal's matrix, but a matrix of higher order derived out of it.

Transformation is as follows:

$$c_{11} = \tilde{A}_1 \cdot \widetilde{B}_1 = [0 \quad a_{11} \quad a_{12} \quad x] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y \\ b_{11} \\ b_{21} \\ 0 \end{bmatrix}$$

Therefore,

$$c_{11} = [(a_{11} + a_{12}) \quad (a_{11} + x) \quad (a_{12} - x) \quad x] \cdot [y \quad (-y + b_{11}) \quad (-y + b_{21}) \quad (-b_{11} + b_{21})]^T$$

Substituting  $x = a_{22}$  and  $y = -b_{22}$ , we get:

$$c_{11} = [(a_{11} + a_{12}) \quad (a_{11} + a_{22}) \quad (a_{12} - a_{22}) \quad a_{22}] \cdot [-b_{22} \quad (b_{22} + b_{11}) \quad (b_{22} + b_{21}) \quad (-b_{11} + b_{21})]^T$$

Calculating  $c_{11}$  and using simultaneous rotation of both indices for  $c_{11}$ , we arrive at:

$$\begin{aligned}
 c_{11} &= [ -(a_{11} + a_{12})b_{22} + (a_{11} + a_{22})(b_{22} + b_{11}) + (a_{12} - a_{22})(b_{22} + b_{21}) + a_{22}(-b_{11} + b_{21}) \\
 &\quad ] \\
 c_{22} &= [ -(a_{22} + a_{21})b_{11} + (a_{22} + a_{11})(b_{11} + b_{22}) + (a_{21} - a_{11})(b_{11} + b_{12}) + a_{11}(-b_{22} + b_{12}) \\
 &\quad ]
 \end{aligned}$$

### Observations

We can observe the fact that all computations of  $c_{i,i+1}$  are reused in the computations of  $c_{i,i}$ , and that there was *one symmetric* computation. This reduced the number of multiplications from 8 to 7. The order of the transformations also do no matter, i.e transforming for  $c_{i,i}$  before  $c_{i,i+1}$  would also have yielded the same reduction in computation. In this case, appropriate choice of *free variables*  $x$  and  $y$ , needs to be done. Furthermore, the placement of the *free variables* also does not matter for second order transform, as long as the dot-product constraints are satisfied. From this exercise, we can safely say that the computation set of  $\{c_{12}, c_{21}\}$ , is *embedded* in the computation set of  $\{c_{11}, c_{22}\}$ . And the size of the second computation set is 7. If this is the smallest such set, then the number must be optimal too.

### Future Directions for budding researchers

1. A natural question to ask is, can we extend these ideas to  $3 \times 3$  and higher-order matrix multiplication?
2. Can we "translate" the derivation here to the language of Group Theory?
3. If we do not find an "optimal embedding" for higher-order matrix multiplication, can we find an approximate embedding?
4. Can invariances, symmetries, embeddings solve one of the most exciting problems in computer science?

### Conclusions

We have systematically derived Strassen's formula for computing  $2 \times 2$  matrix multiplication using 7 multiplications. We modeled matrix multiplication as a vector dot-product computation (which is a well studied area). We explored transformations of vectors which kept the dot-product constant. We discovered degrees of freedom, when transforming the vectors to a higher dimension under the dot-product constraints. We utilized transformations and degrees of freedom to hunt for symmetries which reduced computation from 8 to 7.

The derivation presented in this work was deduced by analyzing and working backwards from Strassen's formula. This is interesting in and itself, to be presented in textbooks and problem-solving workshops alongside discussions of the seminal original work. The insights gained in the derivation can be applied in future work, to study its potential application in higher dimension matrix multiplications. The author feels that the generalization of this derivation to higher dimensions is non-trivial and is a worthy research challenge. However, the author is enthusiastic that a generalization, even to a smaller dimension greater than  $2 \times 2$  matrix may offer new insights, which might lead us closer to the holy grail of matrix multiplication research, i.e a step towards  $\tilde{O}(n^2)$  asymptotic complexity.

### Disclaimer


I do not claim to be an expert in the area of matrix multiplication research. However, I love to solve challenging problems, and find this area intellectually appealing. Readers' feedback is highly anticipated and will be deeply appreciated. I apologize in advance for any factual errors in my presentation, and will do my due diligence to correct them, once they are pointed out to me.

Posted by [Unknown](#) at [11:36](#)

No comments:

[Post a comment](#)

Enter your comment...

 Comment as: Alberto Saa (Gr ▼) Sign out

Publish Preview  Notify me

[Newer Post](#)

[Home](#)

Subscribe to: [Post Comments \(Atom\)](#)

Ethereal theme. Powered by [Blogger](#).