A non-commutative algorithm for multiplying 7 x 7 matrices using 250 multiplications

Alexandre Sedoglavic

A non-commutative algorithm for multiplying 7×7 matrices using 250 multiplications

Alexandre.Sedoglavic@univ-lille.fr

August 4, 2017

Abstract

We present a non-commutative algorithm for multiplying 7×7 matrices using 250 multiplications.

1 Introduction

We use the Strassen's matrix multiplication algorithm [7] to *divide* the 7×7 matrix multiplication problem into smaller sub-problems; the use of Smirnov's matrix multiplication algorithms [5, 6] allows to *conquer* a new upper bound on the number of necessary non-commutative multiplications.

Hence, we present a scheme that evaluate the product $P = N \cdot M$:

$$N = \begin{pmatrix} n_{11} & \cdots & n_{17} \\ \vdots & & \vdots \\ n_{71} & \cdots & n_{77} \end{pmatrix}, \; M = \begin{pmatrix} m_{11} & \cdots & m_{17} \\ \vdots & & \vdots \\ m_{71} & \cdots & m_{77} \end{pmatrix}, \; P = \begin{pmatrix} p_{11} & \cdots & p_{17} \\ \vdots & & \vdots \\ p_{71} & \cdots & p_{77} \end{pmatrix}, \quad (1)$$

using 250 multiplications. This algorithm improves slightly the previous known upper bound 258 presented in [1] and likely obtained with the same kind of technics presented in the next section.

2 Divide

For any 2×2 matrices:

$$A = (a_{ij})_{1 \leq i,j \leq 2}, \quad B = (b_{ij})_{1 \leq i,j \leq 2} \quad \text{and} \quad C = (c_{ij})_{1 \leq i,j \leq 2}, \quad (2)$$

V. Strassen shows in [7] that the matrix product $C = A \cdot B$ could be computed by performing the following operations:

$$\begin{aligned} t_1 &= (a_{11} + a_{22})(b_{11} + b_{22}), & t_2 &= (a_{12} - a_{22})(b_{21} + b_{22}), \\ t_3 &= (-a_{11} + a_{21})(b_{11} + b_{12}), & t_4 &= (a_{11} + a_{12})b_{22}, \\ t_5 &= a_{11}(b_{12} - b_{22}), \; t_6 = a_{22}(-b_{11} + b_{21}), \; t_7 &= (a_{21} + a_{22})b_{11}, \end{aligned} \quad (3)$$

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} t_1 + t_2 - t_4 + t_6 & t_6 + t_7, \\ t_4 + t_5 & t_1 + t_3 + t_5 - t_7 \end{pmatrix},$$

in the considered coefficients algebra.

To construct our algorithm, we are going to work with the algebra of 4×4 matrices and thus, we have to adapt our inputs P, N, M to that end. So we rewrite the matrices (1) in the following equivalent form:

$$X = \left(\begin{array}{cccc|ccc} x_{11} & x_{12} & x_{13} & 0 & x_{14} & \cdots & x_{17} \\ x_{21} & x_{22} & x_{23} & 0 & x_{24} & \cdots & x_{27} \\ x_{31} & x_{32} & x_{33} & 0 & x_{34} & \cdots & x_{37} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \hline x_{41} & x_{42} & x_{34} & 0 & x_{44} & \cdots & x_{34} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ x_{71} & x_{72} & x_{77} & 0 & x_{74} & \cdots & x_{77} \end{array}\right), \quad X \in \{P, M, N\}, \tag{4}$$

in which we have just added a line and a column of zeros. After that padding, the product $P = N \cdot M$ is unchanged.

Notations 2.1 — *Hence, in the sequel P (resp. M, N) designates the 8×8 matrices defined in (4) and P_{ij} (resp. M_{ij}, N_{ij}) designates 4×4 matrices (e.g. P_{11} stands for the upper left submatrix of P, etc).*

Remark 2.1 — *The process of pealing (removing rows and columns) the result of our computations might be better understood if we use the classical equivalence between the bilinear application $\mathcal{B} : \mathbb{K}^{8\times8} \times \mathbb{K}^{8\times8} \mapsto \mathbb{K}^{8\times8}$ with indeterminates N and M that defines the matrix multiplication $\mathcal{B}(N, M) = N \cdot M = P$ and the trilinear form $\mathbb{K}^{8\times8} \times \mathbb{K}^{8\times8} \times \mathbb{K}^{8\times8} \mapsto \mathbb{K}$ with indeterminates N, M and P defined by* $\mathrm{Trace}\,(N \cdot M \cdot P)$ *(see [4, § 2.5.2] and [2] for a complete description of this equivalence).*

Computationally, this equivalence induces that the following relation holds:

$$\begin{aligned} \mathrm{Trace}\,(N \cdot M \cdot P) = {} & \mathrm{Trace}\big(\,(N_{11} + N_{22}) \cdot (M_{11} + M_{22}) \cdot (P_{11} + P_{22})\,\big) && \text{(5a)} \\ & + \mathrm{Trace}\big(\,(N_{12} - N_{22}) \cdot (M_{21} + M_{22}) \cdot P_{11}\big) && \text{(5b)} \\ & + \mathrm{Trace}\big(\,(-N_{11} + N_{21}) \cdot (M_{11} + M_{12}) \cdot P_{22}\big) && \text{(5c)} \\ & + \mathrm{Trace}\big(\,(N_{11} + N_{12}) \cdot M_{22} \cdot (-P_{11} + P_{21})\,\big) && \text{(5d)} \\ & + \mathrm{Trace}\big(N_{11} \cdot (M_{12} - M_{22}) \cdot (P_{21} + P_{22})\,\big) && \text{(5e)} \\ & + \mathrm{Trace}\big(N_{22} \cdot (-M_{11} + M_{21}) \cdot (P_{11} + P_{12})\,\big) && \text{(5f)} \\ & + \mathrm{Trace}\big(\,(N_{21} + N_{22}) \cdot M_{11} \cdot (P_{12} - P_{22})\,\big) && \text{(5g)} \end{aligned}$$

and as the bilinear application and the trilinear form are equivalent, one could retrieve directly the algorithm from this last form. Our original problem is now divided in 7 lower dimensional subproblems encoded by trilinear forms. In the next section, we presents the algorithms used to compute the matrix products (5a-5g).

3 Conquer

The first summand (5a) involves unstructured 4×4 matrix multiplication that could be computed using Strassen's algorithm and could be done with 7^2 multiplications. Before studying the other summands, we emphasise the following trivial remarks:

Remarks 3.1 — *As we consider some* 4×4 *matrices with zero last column and/or row, recall that the product of two* 4×4 *matrices* $(x_{ij})_{1 \leq i,j \leq 4}$ *and matrix* $(y_{ij})_{1 \leq i,j \leq 4}$ *is equivalent to the product of the:*

- 4×3 *matrix* $(x_{ij})_{1 \leq i \leq 4, 1 \leq j \leq 3}$ *with the* 3×4 *matrix* $(y_{ij})_{1 \leq i \leq 3, 1 \leq, j \leq 4}$ *when for* $1 \leq i \leq 4$ *we have* $y_{i4} = 0$ *(zero last row);*
- 3×4 *matrix* $(x_{ij})_{1 \leq i \leq 3, 1 \leq j \leq 4}$ *with the* 4×3 *matrix* $(y_{ij})_{1 \leq i \leq 3, 1 \leq, j \leq 4}$ *when for* $1 \leq j \leq 4$ *we have* $y_{4j} = 0$ *(zero last column).*

Let us now review the matrices involved in the summands (5b-5g).

Remarks 3.2 — *We notice that:*

- *the last row and column of* X_{11} *are only composed by zeros;*
- X_{22} *is a* 4×4 *matrix;*
- *the last column of* $-X_{11} + X_{21}$ *is only composed by zeros;*
- *the last line of* $X_{11} + X_{12}$ *is only composed by zeros;*
- $X_{12} - X_{22}$ *and* $X_{21} - X_{22}$ *are* 4×4 *matrices without zero row or column.*

Hence, taking into account Remarks 3.1 and 3.2, the summand:

- (5b) involves the product of a 3×4 by a 4×3 by a 3×3 matrix;
- (5c) involves the product of 4×3 by 3×4 by 4×4 matrix;
- (5d) involves the product of 3×4 by 4×4 by 4×3 matrix;
- (5e) involves the product of 3×3 by 3×4 by 4×3 matrix;
- (5f) involves the product of 4×4 by 4×3 by 3×4 matrix;
- (5g) involves the product of 4×3 by 3×3 by 3×4 matrix.

Remark 3.3 — *We rely our construction on the representation of matrix multiplication algorithm by trilinear forms and the underlying tensor representation because, as quoted in [3, § 4.6.4 p. 507]:*

> *"For example, a normal scheme for evaluating an* $m \times n$ *times* $n \times s$ *matrix product implies the existence of a normal scheme to evaluate an* $n \times s$ *times* $s \times m$ *matrix product using the same number of chain multiplications."*

This is exactly what we are using in the sequel. To do so and in order to evaluate the summands (5b-5g), we introduce the following notations:

Notation 3.1 — *For any matrices* U, V *and* W *of size* $u \times v, v \times w$ *and* $w \times u$*, we denote by* $\langle u, v, w \rangle$ *the known supremum on the multiplication necessary for computing* $\mathrm{Trace}\,(U \cdot V \cdot W)$ *(that is the number of multiplication used by the best known algorithm allowing to compute* $U \cdot V$ *(a.k.a. tensor rank)).*

Using this notation, we see that:

- (5b) can be computed using $\langle 3, 4, 3\rangle$ multiplications;
- (5c) can be computed using $\langle 4, 3, 4\rangle$ multiplications;
- (5d) can be computed using $\langle 3, 4, 4\rangle$ multiplications;
- (5e) can be computed using $\langle 3, 3, 4\rangle$ multiplications;
- (5f) can be computed using $\langle 4, 4, 3\rangle$ multiplications;
- (5g) can be computed using $\langle 4, 3, 3\rangle$ multiplications.

The remark 3.3 is a direct consequence of the Trace property:

$$\text{Trace}\,(U \cdot V \cdot W) = \text{Trace}\,(W \cdot U \cdot V) = \text{Trace}\,(V \cdot W \cdot U). \tag{6}$$

To be more precise, we recall the following well-known result:

Lemma 3.1 *With the notations 3.1, the following relations hold:*

$$\langle u, v, w\rangle = \langle w, u, v\rangle = \langle v, w, u\rangle. \tag{7}$$

This allows us to state that algorithm (5b-5g) requires:

$$\langle 4, 4, 4\rangle + 3\,\langle 3, 3, 4\rangle + 3\,\langle 3, 4, 4\rangle \tag{8}$$

multiplications. As A. V. Smirnov states that $\langle 3, 3, 4\rangle = 29$ and $\langle 3, 4, 4\rangle = 38$ in [5, Table 1, N$^{\underline{o}}$ 13 and 21], we conclude that our algorithm required 250 multiplications ($7^2 + 3 \cdot 29 + 3 \cdot 38$). Furthermore, Smirnov provides in [6] the explicit description of these algorithms, allowing us to do the same with the algorithm constructed in this note at url:

http://www.fil.univ-lille1.fr/ sedoglav/MUL7x7x7250.txt.

References

[1] Drevet, C.-É., Nazrul Islam, M., and Schost, É. Optimization techniques for small matrix multiplication. *Theoretical Computer Science 412*, 22 (May 2011), 2219–2236.

[2] Dumas, J.-G., and Pan, V. Y. Fast matrix multiplication and symbolic computation. Tech. rep. 1612.05766, arXiv, Dec. 2016.

[3] Knuth, D. E. *The Art of Computer Programming. Seminumerical Algorithms*, 3 ed., vol. 2 of *Computer Science and Information Processing*. Addison Wesley, Reading, Mass., 1997.

[4] Landsberg, J. M. *Tensors: geometry and applications*, vol. 128 of *Graduate Studies in Mathematics*. American Mathematical Society, 2010.

[5] Smirnov, A. V. The bilinear complexity and practical algorithms for matrix multiplication. *Computational Mathematics and Mathemacital Physics 53*, 2 (Dec. 2013), 1781–1795.

[6] Smirnov, A. V. Several bilinear algorithms for matrix multiplication. Tech. rep., ResearchGate, doi: 10.13140/RG.2.2.30005.06886, Jan. 2017.

[7] Strassen, V. Gaussian elimination is not optimal. *Numerische Mathematik 13*, 4 (Aug. 1969), 354–356.