

Fast Commutative Matrix Algorithm

Andreas Rosowski
University of Siegen, Germany
andreas.rosowski@student.uni-siegen.de

July 28, 2020

Abstract

We show that the product of an $n \times 3$ matrix and a 3×3 matrix over a commutative ring can be computed using $6n+3$ multiplications. For two 3×3 matrices this gives us an algorithm using 21 multiplications. This is an improvement with respect to Makarov's algorithm using 22 multiplications [13]. We generalize our result for $n \times 3$ and 3×3 matrices and present an algorithm for computing the product of an $l \times n$ matrix and an $n \times m$ matrix over a commutative ring for odd n using $n(lm + l + m - 1)/2$ multiplications if m is odd and using $(n(lm + l + m - 1) + l - 1)/2$ multiplications if m is even. Waksman's and Islam's algorithm for odd n needs $(n - 1)(lm + l + m - 1)/2 + lm$ multiplications [10, 19], thus in both cases less multiplications are required by our algorithm. We also give an algorithm for even n using $n(lm + l + m - 1)/2$ multiplications without making use of divisions, since Waksman's and Islam's algorithm make use of some divisions by 2 [10, 19]. Furthermore we present a novelty for matrix multiplication: In this paper we show that some non-bilinear algorithms with special properties can be used as recursive algorithms. In comparison to bilinear algorithms for small $n \times n$ matrices say $n < 20$ we obtain some better results. From these non-bilinear algorithms we finally obtain approximate non-bilinear algorithms. For instance we obtain an approximate non-bilinear algorithm for 5×5 matrices that uses only 89 multiplications. If at all it is possible to compare this algorithm with a bilinear algorithm we obtain a better result with respect to Smirnov's algorithm [15].

1 Introduction

In 1969 Strassen showed that the product of two $n \times n$ matrices can be computed using $O(n^{\log_2(7)})$ arithmetic operations [18]. This work opened a new field of research and over the years better upper bounds for the exponent of matrix multiplication were published. In 1990 Coppersmith and Winograd obtained an upper bound of 2.375477 for the exponent [4]. For a long time this was the best result. Since 2010 further improvements were obtained in a series of papers [5, 12, 17, 20, 21]. The best result so far was published in 2014 by Le Gall who obtained an upper bound of 2.3728639 for the exponent [12].

In this paper we first study the product of an $n \times 3$ matrix A and a 3×3 matrix B over a commutative ring and show that we can compute the product AB using $6n + 3$ multiplications. The basic idea is to improve the computation of the product of a 1×3 vector a and a 3×3 matrix B over a commutative ring in the sense that we try to obtain as much as possible multiplications that contain only entries of the matrix B but without using more than 9 multiplications overall. The multiplications which contain only entries of the matrix B only need to be calculated once and can therefore be reused in the matrix multiplication. In the special case $n = 3$ we obtain an algorithm for multiplying two 3×3 matrices using 21 multiplications which improves the best result so far from Makarov using 22 multiplications [13]. Our next step is to generalize this result to the computation of the product of an $l \times n$ matrix A and an $n \times m$ matrix B over a commutative ring for odd n . We show that the product AB can be computed using $n(lm + l + m - 1)/2$ multiplications if m is odd and $(n(lm + l + m - 1) + l - 1)/2$ multiplications if m is even. This improves

Waksman's and Islam's algorithm which require $(n-1)(lm+l+m-1)/2+lm$ multiplications for odd n [10, 19]. Additionally we present an algorithm to multiply an $l \times n$ matrix and an $n \times m$ matrix using $n(lm+l+m-1)/2$ multiplications without making use of divisions, whereas Waksman's and Islam's algorithm makes use of some divisions by 2 [10, 19].

Finally we present a novelty for matrix multiplication. We can show that a non-bilinear algorithm for matrix multiplication can be applied recursively under special circumstances. For this we first generalize the term of bilinear algorithms. Bilinear algorithms have the property that every multiplication is in bilinear form. For the computation of a matrix product AB this means that in every product that is computed the left (resp., right) factor is a sum of entries from A (resp., B). The consequence is that the linear combinations of the products are also in bilinear form. We have found algorithms whose multiplications are not in bilinear form but the linear combinations of the products are in bilinear form. Such an algorithm can be seen as a generalization of bilinear algorithms if one defines a bilinear algorithm independent from the form of the multiplications as an algorithm whose linear combinations are in bilinear form. But in order to distinguish such algorithms from usual bilinear algorithms we call such an algorithm non-bilinear. Of course a non-bilinear algorithm can be applied recursively. Note that by this definition every bilinear algorithm is also a non-bilinear algorithm. We have found formulas for the computation of the product of two $n \times n$ matrices. For even n the algorithm needs $n(n^2+3n+1)/2$ multiplications and $n(n^2+3n+2)/2$ multiplications are required for odd n . In the case $n=14$ this gives an algorithm using 1673 multiplications and an exponent of roughly 2.8125.

Additionally we present approximate non-bilinear algorithm. Approximate algorithms for matrix multiplication were introduced for the first time by Bini et al. [2]. We give a general formula for the computation of the product of two $n \times n$ matrices. For even n the algorithm needs $n(n^2+2n)/2$ multiplications and $n(n^2+2n+1)/2$ multiplications are required for odd n . In the case $n=5$ we were even able to find an approximate non-bilinear algorithm that uses only 89 multiplications.

1.1 Related Work

In this section we present some related work. We start with presenting some results about multiplication of two square matrices. Since Strassen showed in 1969 that the product of two matrices can be computed using $O(n^{\log_2(7)})$ arithmetic operations [18] and since it is shown that for 2×2 matrices 7 is the optimal number of multiplications [8, 22], it is interesting to study $n \times n$ matrices for $n \geq 3$, to obtain an even faster algorithm for matrix multiplication. For 3×3 matrices 21 multiplications would be needed to obtain an even faster algorithm than Strassen's since $\log_3(21) \approx 2.7712 < 2.807 \approx \log_2(7)$. In 1976 Laderman obtained a non-commutative 3×3 algorithm using only 23 multiplications [11]. It is not known if there exists a non-commutative 3×3 algorithm that uses 22 or less multiplications.

Hopcroft and Musinski showed in [9] that the number of multiplications of non-commutative algorithms to compute the product of an $l \times n$ matrix and an $n \times m$ matrix is the same number that is required to compute the product of an $n \times m$ matrix and an $m \times l$ matrix and of an $l \times m$ matrix and an $m \times n$ matrix etc. This means if one finds an algorithm for the product of an $l \times m$ matrix and an $m \times n$ matrix using x multiplications there exists a matrix multiplication algorithm for $lnm \times lnm$ matrices using x^3 multiplications overall. This algorithm for square matrices will then have an exponent of $\log_{lnm}(x^3)$.

We present some examples of non-square matrix multiplication algorithms. In [7] Hopcroft and Kerr showed that the product of a $p \times 2$ matrix and a $2 \times n$ matrix can be multiplied using $\lceil (3pn + \max\{n, p\})/2 \rceil$ multiplications without using commutativity. In the case $p=3=n$ this gives an algorithm using 15 multiplications. Combined with the results of [9] this gives an algorithm for 18×18 matrices using $15^3 = 3375$ multiplications and an exponent of $\log_{18}(3375) \approx 2.811$. Smirnov obtained an algorithm for the product of a 3×3 matrix and a 3×6 matrix using 40 multiplications [16]. By [9] this gives an algorithm for 54×54 matrices using $40^3 = 64000$ multiplications and an exponent of $\log_{54}(64000) \approx 2.7743$.

Cariow et al. developed a high-speed parallel 3×3 matrix multiplier structure based on the commutative 3×3 matrix algorithm using 22 multiplications obtained by Makarov [1, 13]. We suppose that the structure could be improved by using our commutative 3×3 matrix algorithm using 21 multiplications.

In [6] Drevet et al. optimized the number of required multiplications of small matrices up to 30×30 matrices. They considered non-commutative and commutative algorithms. Combined with our results for commutative rings we suppose that some results could be improved.

2 Matrix Multiplication over a Commutative Ring

Let R denote a commutative ring throughout this Section.

2.1 Product of $n \times 3$ and 3×3 Matrices

Consider the product of an 1×3 vector a and a 3×3 matrix B over a commutative ring.

$$a = [a_1 \quad a_2 \quad a_3] \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \quad (1)$$

In the usual way the product of a and B would be computed as:

$$aB = [a_1b_{11} + a_2b_{21} + a_3b_{31} \quad a_1b_{12} + a_2b_{22} + a_3b_{32} \quad a_1b_{13} + a_2b_{23} + a_3b_{33}]$$

But it can also be computed by first computing these 9 products:

Algorithm 1. *Input: Vector a and Matrix B as in (1).*

Let

$$\begin{aligned} p_1 &:= (a_2 + b_{12})(a_1 + b_{21}) \\ p_2 &:= (a_3 + b_{13})(a_1 + b_{31}) \\ p_3 &:= (a_3 + b_{23})(a_2 + b_{32}) \\ p_4 &:= a_1(b_{11} - b_{12} - b_{13} - a_2 - a_3) \\ p_5 &:= a_2(b_{22} - b_{21} - b_{23} - a_1 - a_3) \\ p_6 &:= a_3(b_{33} - b_{31} - b_{32} - a_1 - a_2) \\ p_7 &:= b_{12}b_{21} \\ p_8 &:= b_{13}b_{31} \\ p_9 &:= b_{23}b_{32} \end{aligned}$$

Output:

$$aB = [p_4 + p_1 + p_2 - p_7 - p_8 \quad p_5 + p_1 + p_3 - p_7 - p_9 \quad p_6 + p_2 + p_3 - p_8 - p_9]$$

Theorem 1. *Let R be a commutative ring, let $n \geq 1$, let A be an $n \times 3$ matrix over R and let B be a 3×3 matrix over R . Then the product AB can be computed using $6n + 3$ multiplications.*

Proof. Consider Algorithm 1. The products p_7 , p_8 and p_9 contain only entries of the matrix B . One can observe that for all $n \geq 1$ the products p_7 , p_8 and p_9 can be reused for the product AB and therefore $3(n - 1)$ multiplications are saved. \square

We give an example. In the case $n = 3$ we obtain an algorithm with 21 multiplications for the matrix multiplication. This algorithm needs one multiplication less than Makarov's [13].

Corollary 1. *Let R be a commutative ring and let A and B be 3×3 matrices over R as shown below. Then the product AB can be computed using 21 multiplications as follows:*

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$\begin{aligned} p_1 &:= (a_{12} + b_{12})(a_{11} + b_{21}) \\ p_2 &:= (a_{13} + b_{13})(a_{11} + b_{31}) \\ p_3 &:= (a_{13} + b_{23})(a_{12} + b_{32}) \\ p_4 &:= a_{11}(b_{11} - b_{12} - b_{13} - a_{12} - a_{13}) \\ p_5 &:= a_{12}(b_{22} - b_{21} - b_{23} - a_{11} - a_{13}) \\ p_6 &:= a_{13}(b_{33} - b_{31} - b_{32} - a_{11} - a_{12}) \end{aligned}$$

$$\begin{aligned} p_7 &:= (a_{22} + b_{12})(a_{21} + b_{21}) \\ p_8 &:= (a_{23} + b_{13})(a_{21} + b_{31}) \\ p_9 &:= (a_{23} + b_{23})(a_{22} + b_{32}) \\ p_{10} &:= a_{21}(b_{11} - b_{12} - b_{13} - a_{22} - a_{23}) \\ p_{11} &:= a_{22}(b_{22} - b_{21} - b_{23} - a_{21} - a_{23}) \\ p_{12} &:= a_{23}(b_{33} - b_{31} - b_{32} - a_{21} - a_{22}) \end{aligned}$$

$$\begin{aligned} p_{13} &:= (a_{32} + b_{12})(a_{31} + b_{21}) \\ p_{14} &:= (a_{33} + b_{13})(a_{31} + b_{31}) \\ p_{15} &:= (a_{33} + b_{23})(a_{32} + b_{32}) \\ p_{16} &:= a_{31}(b_{11} - b_{12} - b_{13} - a_{32} - a_{33}) \\ p_{17} &:= a_{32}(b_{22} - b_{21} - b_{23} - a_{31} - a_{33}) \\ p_{18} &:= a_{33}(b_{33} - b_{31} - b_{32} - a_{31} - a_{32}) \end{aligned}$$

$$\begin{aligned} p_{19} &:= b_{12}b_{21} \\ p_{20} &:= b_{13}b_{31} \\ p_{21} &:= b_{23}b_{32} \end{aligned}$$

Hence,

$$AB = \begin{bmatrix} p_4 + p_1 + p_2 - p_{19} - p_{20} & p_5 + p_1 + p_3 - p_{19} - p_{21} & p_6 + p_2 + p_3 - p_{20} - p_{21} \\ p_{10} + p_7 + p_8 - p_{19} - p_{20} & p_{11} + p_7 + p_9 - p_{19} - p_{21} & p_{12} + p_8 + p_9 - p_{20} - p_{21} \\ p_{16} + p_{13} + p_{14} - p_{19} - p_{20} & p_{17} + p_{13} + p_{15} - p_{19} - p_{21} & p_{18} + p_{14} + p_{15} - p_{20} - p_{21} \end{bmatrix}$$

2.2 General Matrix Multiplication

We start this Section with presenting an algorithm for the product of $l \times n$ and $n \times m$ matrices over a commutative ring for even n using $n(lm + l + m - 1)/2$ multiplications. In the literature there is already an algorithm for such matrices with $n(lm + l + m - 1)/2$ multiplications [10, 19], but this algorithm makes use of some divisions by 2. In this Section we present an algorithm that does not use divisions.

Theorem 2. *Let R be a commutative ring, let $n \geq 2$ be even and let $l, m \geq 1$. Furthermore let $A \in R^{l \times n}$, $B \in R^{n \times m}$ be matrices. Then the product AB can be computed using $n(lm + l + m - 1)/2$ multiplications without making use of divisions.*

Proof. Let A and B be matrices as in the Theorem. Let a_{ij} denote the entries of A and let b_{ij} denote the entries of B and let c_{ij} denote the entries of AB . The product AB can be computed as follows. For $i = 1, \dots, l$ let

$$c_{i1} = \sum_{k=1}^{n/2} a_{i(2k-1)}(b_{(2k-1)1} + a_{i(2k)}) + \sum_{k=1}^{n/2} a_{i(2k)}(b_{(2k)1} - a_{i(2k-1)})$$

and for $i = 1, \dots, l$ and $j = 2, \dots, m$ let

$$c_{ij} = \sum_{k=1}^{n/2} (a_{i(2k-1)} + b_{(2k)j})(a_{i(2k)} + b_{(2k-1)1} + b_{(2k-1)j}) - \sum_{k=1}^{n/2} a_{i(2k-1)}(b_{(2k-1)1} + a_{i(2k)}) - \sum_{k=1}^{n/2} b_{(2k)j}(b_{(2k-1)1} + b_{(2k-1)j})$$

These formulas can easily be verified. It remains to show that only $n(lm + l + m - 1)/2$ multiplications are required by these formulas. To show the number of multiplications we count the multiplications. Since the index i takes l different values, the index j takes $m - 1$ different values and the index k takes $n/2$ different values there are $ln/2$ multiplications of the form $a_{i(2k-1)}(b_{(2k-1)1} + a_{i(2k)})$, $ln/2$ multiplications of the form $a_{i(2k)}(b_{(2k)1} - a_{i(2k-1)})$, $n(m - 1)/2$ multiplications of the form $b_{(2k)j}(b_{(2k-1)1} + b_{(2k-1)j})$ and $ln(m - 1)/2$ multiplications of the form $(a_{i(2k-1)} + b_{(2k)j})(a_{i(2k)} + b_{(2k-1)1} + b_{(2k-1)j})$. Thus, overall $ln/2 + ln/2 + n(m - 1)/2 + ln(m - 1)/2 = n(lm + l + m - 1)/2$ multiplications are required. \square

Algorithm 1 from Section 2.1 is the basic idea of a general algorithm for the matrix product of $l \times n$ and $n \times m$ matrices over a commutative ring for odd n . The algorithm we present below is split into two cases. In Case 1 m is odd and in Case 2 m is even. This leads us to the following:

Theorem 3. *Let R be a commutative ring, let $n \geq 3$ be odd, $l \geq 1$, $m \geq 3$ and let $A \in R^{l \times n}$, $B \in R^{n \times m}$ be matrices. Then the following holds:*

- *If m is odd the product AB can be computed using $n(lm + l + m - 1)/2$ multiplications.*
- *If m is even the product AB can be computed using $(n(lm + l + m - 1) + l - 1)/2$ multiplications.*

Proof. Let A and B be matrices as in the Theorem. Now split A and B in submatrices in the following way:

$$A = [A_1 \quad A_2], \text{ with } A_1 \in R^{l \times 3} \text{ and } A_2 \in R^{l \times n-3},$$

$$B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \text{ with } B_1 \in R^{3 \times m} \text{ and } B_2 \in R^{n-3 \times m}.$$

Then $AB = A_1B_1 + A_2B_2$. By using Theorem 2 the product A_2B_2 can be computed using $(n - 3)(lm + l + m - 1)/2$ multiplications. Let a_{ij} denote the entries of A_1 and let b_{ij} denote the entries of B_1 and let c_{ij} denote the entries of A_1B_1 . The matrix A_1B_1 can be computed as follows.

Case 1: m is odd.

For $i = 1, \dots, l$ let

$$c_{i1} = (a_{i1} + b_{21})(a_{i2} + b_{12}) + (a_{i1} + b_{31})(a_{i3} + b_{13}) + a_{i1}(b_{11} - b_{12} - b_{13} - a_{i2} - a_{i3}) - b_{12}b_{21} - b_{13}b_{31}$$

$$c_{i2} = (a_{i1} + b_{21})(a_{i2} + b_{12}) + (a_{i2} + b_{32})(a_{i3} + b_{23}) + a_{i2}(b_{22} - b_{21} - b_{23} - a_{i1} - a_{i3}) - b_{12}b_{21} - b_{23}b_{32}$$

$$c_{i3} = (a_{i1} + b_{31})(a_{i3} + b_{13}) + (a_{i2} + b_{32})(a_{i3} + b_{23}) + a_{i3}(b_{33} - b_{31} - b_{32} - a_{i1} - a_{i2}) - b_{13}b_{31} - b_{23}b_{32}$$

and for $i = 1, \dots, l$ and $j = 4, 6, 8, \dots, m - 1$ let

$$\begin{aligned} c_{ij} &= (a_{i1} + b_{21})(a_{i2} + b_{12}) + (a_{i1} + b_{31})(a_{i3} + b_{13}) + (a_{i1} + b_{21} - b_{2j})(-a_{i2} - b_{12} + b_{1j} - b_{1(j+1)}) \\ &+ (a_{i1} + b_{31} - b_{3j})(-a_{i3} - b_{13} + b_{1(j+1)}) - b_{12}b_{21} - b_{13}b_{31} - (b_{21} - b_{2j})(-b_{12} + b_{1j} - b_{1(j+1)}) \\ &- (b_{31} - b_{3j})(-b_{13} + b_{1(j+1)}) \end{aligned}$$

$$\begin{aligned} c_{i(j+1)} &= (a_{i1} + b_{31})(a_{i3} + b_{13}) + (a_{i2} + b_{32})(a_{i3} + b_{23}) + (a_{i1} + b_{31} - b_{3j})(-a_{i3} - b_{13} + b_{1(j+1)}) \\ &+ (a_{i2} + b_{32} + b_{3j} - b_{3(j+1)})(-a_{i3} - b_{23} + b_{2(j+1)}) - b_{13}b_{31} - b_{23}b_{32} - (b_{31} - b_{3j})(-b_{13} + b_{1(j+1)}) \\ &- (b_{32} + b_{3j} - b_{3(j+1)})(-b_{23} + b_{2(j+1)}) \end{aligned}$$

It can easily be seen that $6l + 3 + 3l(m - 3)/2 + 3(m - 3)/2 = 3(lm + l + m - 1)/2$ multiplications are required to compute A_1B_1 .

Thus, AB can be computed using $3(lm + l + m - 1)/2 + (n - 3)(lm + l + m - 1)/2 = n(lm + l + m - 1)/2$ multiplications.

Case 2: m is even.

For $i = 1, \dots, l$ let

$$c_{i1} = (a_{i1} + b_{21})(a_{i2} + b_{12}) + (a_{i1} + b_{31})(a_{i3} + b_{13}) + a_{i1}(b_{11} - b_{12} - b_{13} - a_{i2} - a_{i3}) - b_{12}b_{21} - b_{13}b_{31}$$

$$c_{i2} = (a_{i1} + b_{21})(a_{i2} + b_{12}) + (a_{i2} + b_{32})(a_{i3} + b_{23}) + a_{i2}(b_{22} - b_{21} - b_{23} - a_{i1} - a_{i3}) - b_{12}b_{21} - b_{23}b_{32}$$

$$c_{i3} = (a_{i1} + b_{31})(a_{i3} + b_{13}) + (a_{i2} + b_{32})(a_{i3} + b_{23}) + a_{i3}(b_{33} - b_{31} - b_{32} - a_{i1} - a_{i2}) - b_{13}b_{31} - b_{23}b_{32}$$

$$c_{i4} = (a_{i1} + b_{21})(a_{i2} + b_{12}) + (a_{i1} + b_{21} - b_{24})(-a_{i2} - b_{12} + b_{14}) + a_{i3}b_{34} - b_{12}b_{21} - (b_{21} - b_{24})(-b_{12} + b_{14})$$

and for $i = 1, \dots, l$ and $j = 5, 7, 9, \dots, m - 1$ let

$$\begin{aligned} c_{ij} &= (a_{i1} + b_{21})(a_{i2} + b_{12}) + (a_{i1} + b_{31})(a_{i3} + b_{13}) + (a_{i1} + b_{21} - b_{2j})(-a_{i2} - b_{12} + b_{1j} - b_{1(j+1)}) \\ &+ (a_{i1} + b_{31} - b_{3j})(-a_{i3} - b_{13} + b_{1(j+1)}) - b_{12}b_{21} - b_{13}b_{31} - (b_{21} - b_{2j})(-b_{12} + b_{1j} - b_{1(j+1)}) \\ &- (b_{31} - b_{3j})(-b_{13} + b_{1(j+1)}) \end{aligned}$$

$$\begin{aligned} c_{i(j+1)} &= (a_{i1} + b_{31})(a_{i3} + b_{13}) + (a_{i2} + b_{32})(a_{i3} + b_{23}) + (a_{i1} + b_{31} - b_{3j})(-a_{i3} - b_{13} + b_{1(j+1)}) \\ &+ (a_{i2} + b_{32} + b_{3j} - b_{3(j+1)})(-a_{i3} - b_{23} + b_{2(j+1)}) - b_{13}b_{31} - b_{23}b_{32} - (b_{31} - b_{3j})(-b_{13} + b_{1(j+1)}) \\ &- (b_{32} + b_{3j} - b_{3(j+1)})(-b_{23} + b_{2(j+1)}) \end{aligned}$$

One can easily verify that in this case $8l + 4 + 3l(m - 4)/2 + 3(m - 4)/2 = 2(l - 1) + 3(lm + m)/2$ multiplications are required to compute A_1B_1 .

Thus, AB can be computed using $2(l - 1) + 3(lm + m)/2 + (n - 3)(lm + l + m - 1)/2 = n(lm + l + m - 1) + l - 1$ multiplications. \square

In both cases less multiplications are required to compute AB than Waksman's and Islam's algorithm [10, 19] for odd n requires.

3 Non-Bilinear Algorithms

Surprisingly it is indeed possible to use a non-bilinear algorithm for matrix multiplication as a recursive algorithm. The crucial point is the form of the results of the linear combinations of the multiplications. The results need to be in bilinear form. Another point is that the identity $AB = BA$ for two $n \times n$ matrices A and B in general does not hold. But for matrices over a commutative ring at least $AB = (B^T A^T)^T$ holds and we will use this fact for our next algorithms.

In this Section we will consider an arbitrary (not necessarily commutative) ring R on which

we define an involution $f : R \rightarrow R$ defined by $A \mapsto A^T$ with the following axioms:

$$(AB)^T = B^T A^T \quad (2)$$

$$(A + B)^T = A^T + B^T$$

In the sense of matrix multiplication A^T denotes the transpose of a matrix A . Note that the following algorithms can be applied recursively if we choose R to be the ring of all square matrices over a commutative ring. Then the entries A_{ij} , B_{ij} and C_{ij} are submatrices.

Theorem 4. *Let R be a ring satisfying (2), let $n \geq 2$ be even. Furthermore let A and B be $n \times n$ matrices over R . Then there is a non-bilinear algorithm using $n(n^2 + 3n + 1)/2$ multiplications for computing the product AB .*

Proof. Let A and B be matrices as in the Theorem. Let A_{ij} denote the entries of A and let B_{ij} denote the entries of B and let C_{ij} denote the entries of AB . The product AB can be computed as follows: For $i = 1, 3, 5, \dots, n-1$ and $j = 1, \dots, n$ let

$$C_{ij} = \sum_{k=1}^n (A_{ik} + B_{jk}^T)(A_{(i+1)j}^T + B_{kj}) - \left(\sum_{k=1}^n A_{ik} + \sum_{k=1}^n B_{jk}^T \right) A_{(i+1)j}^T - \sum_{k=1}^n B_{jk}^T B_{kj}$$

$$C_{(i+1)j} = \sum_{k=1}^n (A_{(i+1)k} + B_{jk}^T)(A_{ij}^T + B_{kj}) - \left(\sum_{k=1}^n A_{(i+1)k} + \sum_{k=1}^n B_{jk}^T \right) A_{ij}^T - \sum_{k=1}^n B_{jk}^T B_{kj}$$

The correctness can easily be verified. It remains to show that only $n(n^2 + 3n + 1)/2$ multiplications are required by these formulas. To show this we first show that for every $l \in \{1, \dots, n\}$ every multiplication of the form $(A_{il} + B_{jl}^T)(A_{(i+1)j}^T + B_{lj})$ is used for the computation of C_{ij} and $C_{(i+1)l}$. It is clear by definition that every multiplication of the form $(A_{il} + B_{jl}^T)(A_{(i+1)j}^T + B_{lj})$ is used for the computation of C_{ij} . We have

$$((A_{il} + B_{jl}^T)(A_{(i+1)j}^T + B_{lj}))^T = (A_{(i+1)j} + B_{lj}^T)(A_{il}^T + B_{jl})$$

Consider the first term in the computation of $C_{(i+1)l}$:

$$\sum_{k=1}^n (A_{(i+1)k} + B_{lk}^T)(A_{il}^T + B_{kl})$$

Since the index k runs over $1, \dots, n$ and $j \in \{1, \dots, n\}$ at some point the index k takes the value j and from this it follows that the multiplications of the form $(A_{il} + B_{jl}^T)(A_{(i+1)j}^T + B_{lj})$ are also used in the computation of $C_{(i+1)l}$. The next step is to count the multiplications. The index i takes $n/2$ different values. The index j takes n different values. Thus overall n^2 multiplications of the form $(\sum_{k=1}^n A_{ik} + \sum_{k=1}^n B_{jk}^T)A_{(i+1)j}^T$ and $(\sum_{k=1}^n A_{(i+1)k} + \sum_{k=1}^n B_{jk}^T)A_{ij}^T$ are computed. Obviously every multiplication of the form $B_{jk}^T B_{kj}$ is used for the computation of C_{ij} and $C_{(i+1)j}$. But $B_{jk}^T B_{kj}$ is also used for the computation of C_{ik} and $C_{(i+1)k}$. Since the multiplications of the form $B_{jk}^T B_{kj}$ are independent from the index i for every i they can be reused. Overall we obtain $n(n-1)/2$ multiplications of the form $B_{jk}^T B_{kj}$ for $k \neq j$ and n multiplications of the form $B_{jk}^T B_{kj}$ for $k = j$ that are computed. Since we have shown that every multiplication of the form $(A_{ik} + B_{jk}^T)(A_{(i+1)j}^T + B_{kj})$ is used for the computation of C_{ij} and $C_{(i+1)k}$ it suffices to count only the multiplications of the form $(A_{(i+1)k} + B_{jk}^T)(A_{ij}^T + B_{kj})$ that are used for the computation of $C_{(i+1)j}$. The index i takes $n/2$ different values, the index j takes n different values and the index k runs over $1, \dots, n$. Hence $n \cdot n \cdot n/2 = n^3/2$ multiplications of the form $(A_{(i+1)k} + B_{jk}^T)(A_{ij}^T + B_{kj})$ are computed. By this the total number of computed multiplications is $2n^2/2 + n(n-1)/2 + n + n^3/2 = n(n^2 + 3n + 1)/2$. \square

Theorem 5. Let R be a ring satisfying (2), let $n \geq 3$ be odd. Furthermore let A and B be $n \times n$ matrices over R . Then there is a non-bilinear algorithm using $n(n^2 + 3n + 2)/2$ multiplications for computing the product AB .

Proof. Let A and B be matrices as in the Theorem. Let A_{ij} denote the entries of A and let B_{ij} denote the entries of B and let C_{ij} denote the entries of AB . The product AB can be computed as follows: For $i = 1, 3, 5, \dots, n - 2$ and $j = 1, \dots, n$ let

$$\begin{aligned} C_{ij} &= \sum_{k=1}^n (A_{ik} + B_{jk}^T)(A_{(i+1)j}^T + B_{kj}) - \left(\sum_{k=1}^n A_{ik} + \sum_{k=1}^n B_{jk}^T \right) A_{(i+1)j}^T - \sum_{k=1}^n B_{jk}^T B_{kj} \\ C_{(i+1)j} &= \sum_{k=1}^n (A_{(i+1)k} + B_{jk}^T)(A_{ij}^T + B_{kj}) - \left(\sum_{k=1}^n A_{(i+1)k} + \sum_{k=1}^n B_{jk}^T \right) A_{ij}^T - \sum_{k=1}^n B_{jk}^T B_{kj} \\ C_{nj} &= \sum_{k=1}^n (A_{nk} + B_{jk}^T)(A_{nj}^T + B_{kj}) - \left(\sum_{k=1}^n A_{nk} + \sum_{k=1}^n B_{jk}^T \right) A_{nj}^T - \sum_{k=1}^n B_{jk}^T B_{kj} \end{aligned}$$

It is easy to verify these terms. The number of computed multiplications can be calculated analogously to Theorem 4. \square

Let us compare the results of Theorems 4 and 5 with an overview of bilinear algorithms:

n	Theorems 4 and 5	[14]
4	58	49
5	105	98
6	165	160
7	252	250
8	356	343
9	495	514
10	655	686
11	858	919
12	1086	1040
13	1365	1443
14	1673	1720
15	2040	2088
16	2440	2401
17	2907	2960
18	3411	3200
19	3990	4065
20	4610	4340

3.1 Optimization

Note that for $k = j$ multiplications of the form $B_{jk}^T B_{kj}$ and $(A_{nk} + B_{jk}^T)(A_{nj}^T + B_{kj})$ can be computed more efficiently. Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Let $C = A^T A$. Consider the computation of matrix C :

$$\begin{aligned} C_{11} &= A_{11}^T A_{11} + A_{21}^T A_{21} \\ C_{12} &= A_{11}^T A_{12} + A_{21}^T A_{22} \\ C_{21} &= A_{12}^T A_{11} + A_{22}^T A_{21} \\ C_{22} &= A_{12}^T A_{12} + A_{22}^T A_{22} \end{aligned}$$

Since we consider computations in a ring satisfying (2) obviously $C_{12}^T = C_{21}$ holds. Thus, $A^T A$ can be computed using 4 recursive calls and 2 multiplications which we can compute with Theorems 4 and 5 again. Now, suppose we are working with matrices over a commutative ring in which $\sqrt{-1}$ exists for instance in the complex numbers. Then $A^T A$ can be computed using only 3 recursive calls and 2 multiplications likewise. To obtain this algorithm we adapted Strassen algorithm [18]:

Theorem 6. *Let R be a ring satisfying (2). Let A be a 2×2 matrix over R and let R' be the underlying ring of the elements of A and let there be an element $x \in R'$ such that $x^2 = -1$. Then the product $A^T A$ can be computed using 5 multiplications.*

Proof. Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Algorithm 2. *Input: Matrix A*

$$\begin{aligned} P_1 &:= (A_{11}^T + \sqrt{-1}A_{22}^T)(A_{11} + \sqrt{-1}A_{22}) \\ P_2 &:= (A_{21}^T + A_{22}^T)(A_{21} + A_{22}) \\ P_3 &:= (A_{11}^T + A_{12}^T)(A_{11} + A_{12}) \\ P_4 &:= (A_{21}^T + \sqrt{-1}A_{11}^T)A_{22} \\ P_5 &:= A_{11}^T(A_{12} - \sqrt{-1}A_{22}) \end{aligned}$$

Output:

$$\begin{aligned} C_{11} &= P_1 + P_2 - P_4 - P_4^T \\ C_{12} &= P_4 + P_5 = C_{21}^T \\ C_{22} &= -P_1 + P_3 - P_5 - P_5^T \end{aligned}$$

□

Note that the products P_1 , P_2 and P_3 are recursive calls. Algorithm 2 is even optimal in the sense that there is no algorithm for computing $A^T A$ that uses 5 multiplications overall from which more than 3 multiplications are recursive calls as the following result shows:

Theorem 7. *Let R be a ring satisfying (2). Let A be a 2×2 matrix over R . Let γ be an algorithm which requires 5 multiplications to compute $A^T A$. Let M_1, \dots, M_5 be the multiplications computed by γ . Let α_i the left factor of M_i and let β_i the right factor of M_i . Then there are at most 3 pairwise different indices $1 \leq i_1 < i_2 < i_3 \leq 5$ such that for $j = 1, 2, 3$ the following holds:*

$$\alpha_{i_j}^T = \beta_{i_j}$$

Proof. Let A be a matrix as in the Theorem. Let γ be an algorithm, which requires 5 multiplications for the computation of $A^T A$. Let M_1, \dots, M_5 be the computed multiplications of γ . Let α_i the left factor of M_i and let β_i the right factor of M_i . Suppose there are 4 pairwise different indices $1 \leq i_1 < i_2 < i_3 < i_4 \leq 5$ such that for $j = 1, 2, 3, 4$ we have $\alpha_{i_j}^T = \beta_{i_j}$. Without loss of generality we choose $i_j = j$. Let

$$E = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix} \quad F = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}$$

We choose a new matrix G as follows:

$$G^T = \begin{bmatrix} E_{11} & E_{12} \\ F_{11}^T & F_{21}^T \\ E_{21} & E_{22} \\ F_{12}^T & F_{22}^T \end{bmatrix}$$

Then the product EF is computed by $G^T G$. We apply γ to compute the product $G^T G$. Since G^T is a 4×2 matrix the multiplications M_1, \dots, M_5 compute products of 2×1 and 1×2 submatrices. For $i = 1, \dots, 4$ we have:

$$M_i = \alpha_i \beta_i = \beta_i^T \beta_i = \begin{bmatrix} \beta_{i1}^T \\ \beta_{i2}^T \end{bmatrix} [\beta_{i1} \quad \beta_{i2}]$$

and

$$M_5 = \alpha_5 \beta_5 = \begin{bmatrix} \alpha_{51} \\ \alpha_{52} \end{bmatrix} [\beta_{51} \quad \beta_{52}]$$

For the computation of the products M_1, \dots, M_5 we need 16 multiplications. For $i = 1, \dots, 4$ let:

$$\begin{aligned} M_{i1} &:= \beta_{i1}^T \beta_{i1} \\ M_{i2} &:= \beta_{i1}^T \beta_{i2} \\ M_{i3} &:= \beta_{i2}^T \beta_{i1} \\ M_{i4} &:= \beta_{i2}^T \beta_{i2} \end{aligned}$$

Furthermore let:

$$\begin{aligned} M_{51} &:= \alpha_{51} \beta_{51} \\ M_{52} &:= \alpha_{51} \beta_{52} \\ M_{53} &:= \alpha_{52} \beta_{51} \\ M_{54} &:= \alpha_{52} \beta_{52} \end{aligned}$$

For $i = 1, \dots, 4$ we have:

$$M_i = \begin{bmatrix} M_{i1} & M_{i2} \\ M_{i3} & M_{i4} \end{bmatrix}$$

Besides:

$$M_5 = \begin{bmatrix} M_{51} & M_{52} \\ M_{53} & M_{54} \end{bmatrix}$$

Let

$$G^T G = D = \begin{bmatrix} D_1 & D_2 \\ D_3 & D_4 \end{bmatrix}$$

be the product computed by γ in which for $i = 1, \dots, 4$ let

$$D_i = \begin{bmatrix} D_{i1} & D_{i2} \\ D_{i3} & D_{i4} \end{bmatrix}$$

It follows:

$$\begin{aligned} D_1 &= \begin{bmatrix} E_{11} & E_{12} \\ F_{11}^T & F_{21}^T \end{bmatrix} \cdot \begin{bmatrix} E_{11}^T & F_{11} \\ E_{12}^T & F_{21} \end{bmatrix} & D_2 &= \begin{bmatrix} E_{11} & E_{12} \\ F_{11}^T & F_{21}^T \end{bmatrix} \cdot \begin{bmatrix} E_{21}^T & F_{12} \\ E_{22}^T & F_{22} \end{bmatrix} \\ D_3 &= \begin{bmatrix} E_{21} & E_{22} \\ F_{12}^T & F_{22}^T \end{bmatrix} \cdot \begin{bmatrix} E_{11}^T & F_{11} \\ E_{12}^T & F_{21} \end{bmatrix} & D_4 &= \begin{bmatrix} E_{21} & E_{22} \\ F_{12}^T & F_{22}^T \end{bmatrix} \cdot \begin{bmatrix} E_{21}^T & F_{12} \\ E_{22}^T & F_{22} \end{bmatrix} \end{aligned}$$

For $k = 1, \dots, 5$ and $i = 1, \dots, 4$ there are constants d_{ki} and e_{ki} such that:

$$D_i = \sum_{k=1}^5 d_{ki} M_k + \sum_{k=1}^5 e_{ki} M_k^T \quad (3)$$

But we are only interested in the subproduct EF . We have:

$$EF = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix} \cdot \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} = \begin{bmatrix} D_{12} & D_{22} \\ D_{32} & D_{42} \end{bmatrix}$$

However, by equation (3) only the products $M_{12}, M_{22}, M_{32}, M_{42}, M_{52}$ and M_{53} come into question for computing the product EF . By this fact we have found an algorithm to compute the product of two 2×2 matrices using only 6 multiplications. However, it was shown that even when using commutativity at least 7 multiplications are required to compute the product of two 2×2 matrices [8, 22]. This is a contradiction. \square

3.2 Approximate algorithms

Approximate algorithms for matrix multiplication were first introduced by Bini et al. [2]. The idea of approximate algorithms for matrix multiplication is to introduce an additional factor usually denoted by ε or λ that allows to compute an error term in addition to the correct product. Letting ε tend to 0 allows to compute the product with any accuracy. In addition approximate algorithms can be used to obtain an exact result. The interested reader is referred to [3].

From the Theorems 4 and 5 we can easily obtain approximate algorithms.

Theorem 8. *Let R be a ring satisfying (2), let $n \geq 2$ be even. Furthermore let A and B be $n \times n$ matrices over R . Then there is an approximate non-bilinear algorithm using $n(n^2 + 2n)/2$ multiplications for computing the product AB .*

Proof. Let A and B be matrices as in the Theorem. Let A_{ij} denote the entries of A and let B_{ij} denote the entries of B and let C_{ij} denote the entries of AB . The product AB can be computed as follows: For $i = 1, 3, 5, \dots, n-1$ and $j = 1, \dots, n$ let

$$C_{ij} = \frac{1}{\varepsilon} \left(\sum_{k=1}^n (A_{ik} + B_{jk}^T \varepsilon) (A_{(i+1)j}^T + B_{kj} \varepsilon) - \left(\sum_{k=1}^n A_{ik} + \sum_{k=1}^n B_{jk}^T \varepsilon \right) A_{(i+1)j}^T \right)$$

$$C_{(i+1)j} = \frac{1}{\varepsilon} \left(\sum_{k=1}^n (A_{(i+1)k} + B_{jk}^T \varepsilon) (A_{ij}^T + B_{kj} \varepsilon) - \left(\sum_{k=1}^n A_{(i+1)k} + \sum_{k=1}^n B_{jk}^T \varepsilon \right) A_{ij}^T \right)$$

\square

Theorem 9. *Let R be a ring satisfying (2), let $n \geq 3$ be odd. Furthermore let A and B be $n \times n$ matrices over R . Then there is an approximate non-bilinear algorithm using $n(n^2 + 2n + 1)/2$ multiplications for computing the product AB .*

Proof. Let A and B be matrices as in the Theorem. Let A_{ij} denote the entries of A and let B_{ij} denote the entries of B and let C_{ij} denote the entries of AB . The product AB can be computed as follows: For $i = 1, 3, 5, \dots, n-2$ and $j = 1, \dots, n$ let

$$C_{ij} = \frac{1}{\varepsilon} \left(\sum_{k=1}^n (A_{ik} + B_{jk}^T \varepsilon) (A_{(i+1)j}^T + B_{kj} \varepsilon) - \left(\sum_{k=1}^n A_{ik} + \sum_{k=1}^n B_{jk}^T \varepsilon \right) A_{(i+1)j}^T \right)$$

$$C_{(i+1)j} = \frac{1}{\varepsilon} \left(\sum_{k=1}^n (A_{(i+1)k} + B_{jk}^T \varepsilon) (A_{ij}^T + B_{kj} \varepsilon) - \left(\sum_{k=1}^n A_{(i+1)k} + \sum_{k=1}^n B_{jk}^T \varepsilon \right) A_{ij}^T \right)$$

$$C_{nj} = \frac{1}{\varepsilon} \left(\sum_{k=1}^n (A_{nk} + B_{jk}^T \varepsilon) (A_{nj}^T + B_{kj} \varepsilon) - \left(\sum_{k=1}^n A_{nk} + \sum_{k=1}^n B_{jk}^T \varepsilon \right) A_{nj}^T \right)$$

\square

Note that the algorithms of Theorems 8 and 9 are of degree 1. In the case where $n = 5$ Theorem 9 yields an algorithm using only 90 multiplications. The best approximate bilinear algorithm for 5×5 matrices we are aware of was published by Smirnov [15] and uses also 90 multiplications. But in this special case we were even able to find an approximate non-bilinear algorithm that uses only 89 multiplications. Nevertheless this algorithm is only of degree 2.

Theorem 10. *There is an approximate non-bilinear algorithm that computes the product of two 5×5 matrices with only 89 multiplications.*

Proof. Let

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} \\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} & B_{13} & B_{14} & B_{15} \\ B_{21} & B_{22} & B_{23} & B_{24} & B_{25} \\ B_{31} & B_{32} & B_{33} & B_{34} & B_{35} \\ B_{41} & B_{42} & B_{43} & B_{44} & B_{45} \\ B_{51} & B_{52} & B_{53} & B_{54} & B_{55} \end{bmatrix}$$

Let

$$AB = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} \end{bmatrix}$$

Algorithm 3. *Input: Matrices A and B . Let:*

$$\begin{aligned} M_1 &:= (A_{11} + B_{11}^T \varepsilon^2)(B_{11} \varepsilon^2 + A_{21}^T) & M_6 &:= (A_{11} + B_{21}^T \varepsilon^2)(B_{12} \varepsilon^2 + A_{22}^T) \\ M_2 &:= (A_{12} + B_{12}^T \varepsilon^2)(B_{21} \varepsilon^2 + A_{21}^T) & M_7 &:= (A_{12} + B_{22}^T \varepsilon^2)(B_{22} \varepsilon^2 + A_{22}^T) \\ M_3 &:= (A_{13} + B_{13}^T \varepsilon^2)(B_{31} \varepsilon^2 + A_{21}^T) & M_8 &:= (A_{13} + B_{23}^T \varepsilon^2)(B_{32} \varepsilon^2 + A_{22}^T) \\ M_4 &:= (A_{14} + B_{14}^T \varepsilon^2)(B_{41} \varepsilon^2 + A_{21}^T) & M_9 &:= (A_{14} + B_{24}^T \varepsilon^2)(B_{42} \varepsilon^2 + A_{22}^T) \\ M_5 &:= (A_{15} + B_{15}^T \varepsilon^2)(B_{51} \varepsilon^2 + A_{21}^T) & M_{10} &:= (A_{15} + B_{25}^T \varepsilon^2)(B_{52} \varepsilon^2 + A_{22}^T) \\ \\ M_{11} &:= (A_{11} + B_{31}^T \varepsilon^2)(B_{13} \varepsilon^2 + A_{23}^T) & M_{16} &:= (A_{11} + B_{41}^T \varepsilon^2)(B_{14} \varepsilon^2 + A_{24}^T) \\ M_{12} &:= (A_{12} + B_{32}^T \varepsilon^2)(B_{23} \varepsilon^2 + A_{23}^T) & M_{17} &:= (A_{12} + B_{42}^T \varepsilon^2)(B_{24} \varepsilon^2 + A_{24}^T) \\ M_{13} &:= (A_{13} + B_{33}^T \varepsilon^2)(B_{33} \varepsilon^2 + A_{23}^T) & M_{18} &:= (A_{13} + B_{43}^T \varepsilon^2)(B_{34} \varepsilon^2 + A_{24}^T) \\ M_{14} &:= (A_{14} + B_{34}^T \varepsilon^2)(B_{43} \varepsilon^2 + A_{23}^T) & M_{19} &:= (A_{14} + B_{44}^T \varepsilon^2)(B_{44} \varepsilon^2 + A_{24}^T) \\ M_{15} &:= (A_{15} + B_{35}^T \varepsilon^2)(B_{53} \varepsilon^2 + A_{23}^T) & M_{20} &:= (A_{15} + B_{45}^T \varepsilon^2)(B_{54} \varepsilon^2 + A_{24}^T) \end{aligned}$$

$$\begin{aligned} M_{21} &:= (A_{11} + A_{12} + A_{13} + A_{14} + A_{15} + B_{11}^T \varepsilon^2 + B_{12}^T \varepsilon^2 + B_{13}^T \varepsilon^2 + B_{14}^T \varepsilon^2 + B_{15}^T \varepsilon^2) A_{21}^T \\ M_{22} &:= (A_{11} + A_{12} + A_{13} + A_{14} + A_{15} + B_{21}^T \varepsilon^2 + B_{22}^T \varepsilon^2 + B_{23}^T \varepsilon^2 + B_{24}^T \varepsilon^2 + B_{25}^T \varepsilon^2) A_{22}^T \\ M_{23} &:= (A_{11} + A_{12} + A_{13} + A_{14} + A_{15} + B_{31}^T \varepsilon^2 + B_{32}^T \varepsilon^2 + B_{33}^T \varepsilon^2 + B_{34}^T \varepsilon^2 + B_{35}^T \varepsilon^2) A_{23}^T \\ M_{24} &:= (A_{11} + A_{12} + A_{13} + A_{14} + A_{15} + B_{41}^T \varepsilon^2 + B_{42}^T \varepsilon^2 + B_{43}^T \varepsilon^2 + B_{44}^T \varepsilon^2 + B_{45}^T \varepsilon^2) A_{24}^T \end{aligned}$$

$$\begin{aligned} M_{25} &:= (A_{21} + A_{22} + A_{23} + A_{24} + B_{11}^T \varepsilon^2 + B_{12}^T \varepsilon^2 + B_{13}^T \varepsilon^2 + B_{14}^T \varepsilon^2 - B_{15}^T \varepsilon^3) A_{11}^T \\ M_{26} &:= (A_{21} + A_{22} + A_{23} + A_{24} + B_{21}^T \varepsilon^2 + B_{22}^T \varepsilon^2 + B_{23}^T \varepsilon^2 + B_{24}^T \varepsilon^2 - B_{25}^T \varepsilon^3) A_{12}^T \\ M_{27} &:= (A_{21} + A_{22} + A_{23} + A_{24} + B_{31}^T \varepsilon^2 + B_{32}^T \varepsilon^2 + B_{33}^T \varepsilon^2 + B_{34}^T \varepsilon^2 - B_{35}^T \varepsilon^3) A_{13}^T \\ M_{28} &:= (A_{21} + A_{22} + A_{23} + A_{24} + B_{41}^T \varepsilon^2 + B_{42}^T \varepsilon^2 + B_{43}^T \varepsilon^2 + B_{44}^T \varepsilon^2 - B_{45}^T \varepsilon^3) A_{14}^T \\ M_{29} &:= (A_{21} + A_{22} + A_{23} + A_{24} + B_{51}^T \varepsilon^2 + B_{52}^T \varepsilon^2 + B_{53}^T \varepsilon^2 + B_{54}^T \varepsilon^2 - B_{55}^T \varepsilon^3) A_{15}^T \end{aligned}$$

$$M_{30} := (A_{11} + A_{12} + A_{13} + A_{14} + A_{15})(A_{21}^T + A_{22}^T + A_{23}^T + A_{24}^T)$$

$$\begin{aligned} M_{31} &:= (A_{31} + B_{11}^T \varepsilon^2)(B_{11} \varepsilon^2 + A_{41}^T) & M_{36} &:= (A_{31} + B_{21}^T \varepsilon^2)(B_{12} \varepsilon^2 + A_{42}^T) \\ M_{32} &:= (A_{32} + B_{12}^T \varepsilon^2)(B_{21} \varepsilon^2 + A_{41}^T) & M_{37} &:= (A_{32} + B_{22}^T \varepsilon^2)(B_{22} \varepsilon^2 + A_{42}^T) \\ M_{33} &:= (A_{33} + B_{13}^T \varepsilon^2)(B_{31} \varepsilon^2 + A_{41}^T) & M_{38} &:= (A_{33} + B_{23}^T \varepsilon^2)(B_{32} \varepsilon^2 + A_{42}^T) \\ M_{34} &:= (A_{34} + B_{14}^T \varepsilon^2)(B_{41} \varepsilon^2 + A_{41}^T) & M_{39} &:= (A_{34} + B_{24}^T \varepsilon^2)(B_{42} \varepsilon^2 + A_{42}^T) \\ M_{35} &:= (A_{35} + B_{15}^T \varepsilon^2)(B_{51} \varepsilon^2 + A_{41}^T) & M_{40} &:= (A_{35} + B_{25}^T \varepsilon^2)(B_{52} \varepsilon^2 + A_{42}^T) \end{aligned}$$

$$\begin{aligned} M_{41} &:= (A_{31} + B_{31}^T \varepsilon^2)(B_{13} \varepsilon^2 + A_{43}^T) & M_{46} &:= (A_{31} + B_{41}^T \varepsilon^2)(B_{14} \varepsilon^2 + A_{44}^T) \\ M_{42} &:= (A_{32} + B_{32}^T \varepsilon^2)(B_{23} \varepsilon^2 + A_{43}^T) & M_{47} &:= (A_{32} + B_{42}^T \varepsilon^2)(B_{24} \varepsilon^2 + A_{44}^T) \\ M_{43} &:= (A_{33} + B_{33}^T \varepsilon^2)(B_{33} \varepsilon^2 + A_{43}^T) & M_{48} &:= (A_{33} + B_{43}^T \varepsilon^2)(B_{34} \varepsilon^2 + A_{44}^T) \\ M_{44} &:= (A_{34} + B_{34}^T \varepsilon^2)(B_{43} \varepsilon^2 + A_{43}^T) & M_{49} &:= (A_{34} + B_{44}^T \varepsilon^2)(B_{44} \varepsilon^2 + A_{44}^T) \\ M_{45} &:= (A_{35} + B_{35}^T \varepsilon^2)(B_{53} \varepsilon^2 + A_{43}^T) & M_{50} &:= (A_{35} + B_{45}^T \varepsilon^2)(B_{54} \varepsilon^2 + A_{44}^T) \end{aligned}$$

$$\begin{aligned} M_{51} &:= (A_{31} + A_{32} + A_{33} + A_{34} + A_{35} + B_{11}^T \varepsilon^2 + B_{12}^T \varepsilon^2 + B_{13}^T \varepsilon^2 + B_{14}^T \varepsilon^2 + B_{15}^T \varepsilon^2) A_{41}^T \\ M_{52} &:= (A_{31} + A_{32} + A_{33} + A_{34} + A_{35} + B_{21}^T \varepsilon^2 + B_{22}^T \varepsilon^2 + B_{23}^T \varepsilon^2 + B_{24}^T \varepsilon^2 + B_{25}^T \varepsilon^2) A_{42}^T \\ M_{53} &:= (A_{31} + A_{32} + A_{33} + A_{34} + A_{35} + B_{31}^T \varepsilon^2 + B_{32}^T \varepsilon^2 + B_{33}^T \varepsilon^2 + B_{34}^T \varepsilon^2 + B_{35}^T \varepsilon^2) A_{43}^T \\ M_{54} &:= (A_{31} + A_{32} + A_{33} + A_{34} + A_{35} + B_{41}^T \varepsilon^2 + B_{42}^T \varepsilon^2 + B_{43}^T \varepsilon^2 + B_{44}^T \varepsilon^2 + B_{45}^T \varepsilon^2) A_{44}^T \end{aligned}$$

$$\begin{aligned} M_{55} &:= (A_{41} + A_{42} + A_{43} + A_{44} + B_{11}^T \varepsilon^2 + B_{12}^T \varepsilon^2 + B_{13}^T \varepsilon^2 + B_{14}^T \varepsilon^2 - B_{15}^T \varepsilon^3) A_{31}^T \\ M_{56} &:= (A_{41} + A_{42} + A_{43} + A_{44} + B_{21}^T \varepsilon^2 + B_{22}^T \varepsilon^2 + B_{23}^T \varepsilon^2 + B_{24}^T \varepsilon^2 - B_{25}^T \varepsilon^3) A_{32}^T \\ M_{57} &:= (A_{41} + A_{42} + A_{43} + A_{44} + B_{31}^T \varepsilon^2 + B_{32}^T \varepsilon^2 + B_{33}^T \varepsilon^2 + B_{34}^T \varepsilon^2 - B_{35}^T \varepsilon^3) A_{33}^T \\ M_{58} &:= (A_{41} + A_{42} + A_{43} + A_{44} + B_{41}^T \varepsilon^2 + B_{42}^T \varepsilon^2 + B_{43}^T \varepsilon^2 + B_{44}^T \varepsilon^2 - B_{45}^T \varepsilon^3) A_{34}^T \\ M_{59} &:= (A_{41} + A_{42} + A_{43} + A_{44} + B_{51}^T \varepsilon^2 + B_{52}^T \varepsilon^2 + B_{53}^T \varepsilon^2 + B_{54}^T \varepsilon^2 - B_{55}^T \varepsilon^3) A_{35}^T \end{aligned}$$

$$M_{60} := (A_{31} + A_{32} + A_{33} + A_{34} + A_{35})(A_{41}^T + A_{42}^T + A_{43}^T + A_{44}^T)$$

$$\begin{aligned} M_{61} &:= (A_{25} + A_{51} \varepsilon + A_{45})(B_{51} + B_{11} \varepsilon) & M_{66} &:= (A_{51} \varepsilon + A_{45}) B_{51} \\ M_{62} &:= (A_{25} + A_{52} \varepsilon + A_{45})(B_{52} + B_{21} \varepsilon) & M_{67} &:= (A_{52} \varepsilon + A_{45}) B_{52} \\ M_{63} &:= (A_{25} + A_{53} \varepsilon + A_{45})(B_{53} + B_{31} \varepsilon) & M_{68} &:= (A_{53} \varepsilon + A_{45}) B_{53} \\ M_{64} &:= (A_{25} + A_{54} \varepsilon + A_{45})(B_{54} + B_{41} \varepsilon) & M_{69} &:= (A_{54} \varepsilon + A_{45}) B_{54} \\ M_{65} &:= (A_{25} + A_{55} \varepsilon + A_{45})(B_{55} + B_{51} \varepsilon) & M_{70} &:= (A_{55} \varepsilon + A_{45}) B_{55} \end{aligned}$$

$$M_{71} := (A_{25} + A_{45})(B_{51} + B_{52} + B_{53} + B_{54} + B_{55} + B_{11} \varepsilon + B_{21} \varepsilon + B_{31} \varepsilon + B_{41} \varepsilon + B_{51} \varepsilon)$$

$$M_{72} := A_{45}(B_{51} + B_{52} + B_{53} + B_{54} + B_{55})$$

$$\begin{aligned} M_{73} &:= (A_{51} + B_{42}^T \varepsilon^2)(B_{12} \varepsilon^2 + A_{54}^T) & M_{76} &:= (A_{51} + B_{52}^T \varepsilon^2)(B_{13} \varepsilon^2 + A_{55}^T) \\ M_{74} &:= (A_{52} + B_{43}^T \varepsilon^2)(B_{22} \varepsilon^2 + A_{54}^T) & M_{77} &:= (A_{52} + B_{53}^T \varepsilon^2)(B_{23} \varepsilon^2 + A_{55}^T) \\ M_{75} &:= (A_{53} + B_{44}^T \varepsilon^2)(B_{32} \varepsilon^2 + A_{54}^T) & M_{78} &:= (A_{53} + B_{54}^T \varepsilon^2)(B_{33} \varepsilon^2 + A_{55}^T) \end{aligned}$$

$$M_{79} := (A_{51} + A_{52} + A_{53} + B_{42}^T \varepsilon^2 + B_{43}^T \varepsilon^2 + B_{44}^T \varepsilon^2) A_{54}^T$$

$$M_{80} := (A_{51} + A_{52} + A_{53} + B_{52}^T \varepsilon^2 + B_{53}^T \varepsilon^2 + B_{54}^T \varepsilon^2) A_{55}^T$$

$$M_{81} := (A_{54} + A_{55} + B_{12}^T \varepsilon^2 + B_{13}^T \varepsilon^2 - B_{14}^T \varepsilon^3) A_{51}^T$$

$$M_{82} := (A_{54} + A_{55} + B_{22}^T \varepsilon^2 + B_{23}^T \varepsilon^2 - B_{24}^T \varepsilon^3) A_{52}^T$$

$$M_{83} := (A_{54} + A_{55} + B_{32}^T \varepsilon^2 + B_{33}^T \varepsilon^2 - B_{34}^T \varepsilon^3) A_{53}^T$$

$$M_{84} := (A_{51} + A_{52} + A_{53})(A_{54}^T + A_{55}^T)$$

$$M_{85} := A_{51} B_{15}$$

$$M_{86} := A_{52} B_{25}$$

$$M_{87} := A_{53} B_{35}$$

$$M_{88} := A_{54} B_{45}$$

$$M_{89} := A_{55} B_{55}$$

Output:

$$C_{11} = \frac{1}{\varepsilon^2} (M_1 + M_2 + M_3 + M_4 + M_5 - M_{21})$$

$$C_{12} = \frac{1}{\varepsilon^2} (M_6 + M_7 + M_8 + M_9 + M_{10} - M_{22})$$

$$C_{13} = \frac{1}{\varepsilon^2} (M_{11} + M_{12} + M_{13} + M_{14} + M_{15} - M_{23})$$

$$C_{14} = \frac{1}{\varepsilon^2} (M_{16} + M_{17} + M_{18} + M_{19} + M_{20} - M_{24})$$

$$C_{15} = \frac{1}{\varepsilon^3} (M_1 + M_2 + M_3 + M_4 + M_5 + M_6 + M_7 + M_8 + M_9 + M_{10} + M_{11} + M_{12}$$

$$+ M_{13} + M_{14} + M_{15} + M_{16} + M_{17} + M_{18} + M_{19} + M_{20} - M_{21} - M_{22} - M_{23} - M_{24}$$

$$- M_{25}^T - M_{26}^T - M_{27}^T - M_{28}^T - M_{29}^T + M_{30})$$

$$C_{21} = \frac{1}{\varepsilon^2} (M_1^T + M_6^T + M_{11}^T + M_{16}^T - M_{25}) + M_{61} - M_{66}$$

$$C_{22} = \frac{1}{\varepsilon^2} (M_2^T + M_7^T + M_{12}^T + M_{17}^T - M_{26}) + M_{62} - M_{67}$$

$$C_{23} = \frac{1}{\varepsilon^2} (M_3^T + M_8^T + M_{13}^T + M_{18}^T - M_{27}) + M_{63} - M_{68}$$

$$C_{24} = \frac{1}{\varepsilon^2} (M_4^T + M_9^T + M_{14}^T + M_{19}^T - M_{28}) + M_{64} - M_{69}$$

$$C_{25} = \frac{1}{\varepsilon^2} (M_5^T + M_{10}^T + M_{15}^T + M_{20}^T - M_{29}) + M_{65} - M_{70}$$

$$\begin{aligned}
C_{31} &= \frac{1}{\varepsilon^2}(M_{31} + M_{32} + M_{33} + M_{34} + M_{35} - M_{51}) \\
C_{32} &= \frac{1}{\varepsilon^2}(M_{36} + M_{37} + M_{38} + M_{39} + M_{40} - M_{52}) \\
C_{33} &= \frac{1}{\varepsilon^2}(M_{41} + M_{42} + M_{43} + M_{44} + M_{45} - M_{53}) \\
C_{34} &= \frac{1}{\varepsilon^2}(M_{46} + M_{47} + M_{48} + M_{49} + M_{50} - M_{54}) \\
C_{35} &= \frac{1}{\varepsilon^3}(M_{31} + M_{32} + M_{33} + M_{34} + M_{35} + M_{36} + M_{37} + M_{38} + M_{39} + M_{40} + M_{41} + M_{42} \\
&\quad + M_{43} + M_{44} + M_{45} + M_{46} + M_{47} + M_{48} + M_{49} + M_{50} - M_{51} - M_{52} - M_{53} - M_{54} \\
&\quad - M_{55}^T - M_{56}^T - M_{57}^T - M_{58}^T - M_{59}^T + M_{60})
\end{aligned}$$

$$\begin{aligned}
C_{41} &= \frac{1}{\varepsilon^2}(M_{31}^T + M_{36}^T + M_{41}^T + M_{46}^T - M_{55}) + M_{66} \\
C_{42} &= \frac{1}{\varepsilon^2}(M_{32}^T + M_{37}^T + M_{42}^T + M_{47}^T - M_{56}) + M_{67} \\
C_{43} &= \frac{1}{\varepsilon^2}(M_{33}^T + M_{38}^T + M_{43}^T + M_{48}^T - M_{57}) + M_{68} \\
C_{44} &= \frac{1}{\varepsilon^2}(M_{34}^T + M_{39}^T + M_{44}^T + M_{49}^T - M_{58}) + M_{69} \\
C_{45} &= \frac{1}{\varepsilon^2}(M_{35}^T + M_{40}^T + M_{45}^T + M_{50}^T - M_{59}) + M_{70}
\end{aligned}$$

$$\begin{aligned}
C_{51} &= \frac{1}{\varepsilon^2}(M_{61} + M_{62} + M_{63} + M_{64} + M_{65} - M_{66} - M_{67} - M_{68} - M_{69} - M_{70} - M_{71} + M_{72}) \\
C_{52} &= \frac{1}{\varepsilon^2}(M_{73} + M_{74} + M_{75} - M_{79} + M_{73}^T + M_{76}^T - M_{81}) \\
C_{53} &= \frac{1}{\varepsilon^2}(M_{76} + M_{77} + M_{78} - M_{80} + M_{74}^T + M_{77}^T - M_{82}) \\
C_{54} &= \frac{1}{\varepsilon^3}(M_{73} + M_{74} + M_{75} + M_{76} + M_{77} + M_{78} - M_{79} - M_{80} - M_{81}^T - M_{82}^T - M_{83}^T + M_{84} + M_{75}^T\varepsilon + M_{78}^T\varepsilon - M_{83}\varepsilon) \\
C_{55} &= M_{85} + M_{86} + M_{87} + M_{88} + M_{89}
\end{aligned}$$

□

4 Open Problem

We denote the minimal number of multiplications required by a non-bilinear algorithm to compute the product of two $n \times n$ matrices by non-bilinear rank and use $R_{nb}(\langle n, n, n \rangle)$ for that. Since by definition every bilinear algorithm is a non-bilinear algorithm it is clear that

$$R_{nb}(\langle n, n, n \rangle) \leq R(\langle n, n, n \rangle)$$

But it remains open if equality holds.

5 Acknowledgment

I am grateful to Michael Figelius and Markus Lohrey for helpful comments.

References

- [1] A. Cariow, W. Sysło, G. Cariowa, M. Gliszczynski. A rationalized structure of processing unit to multiply 3×3 matrices. *Journal Pomiaru Automatyka Kontrola*, Volume R. 58, Number 7, (2012), 677–680
- [2] D. Bini, M. Capovani, F. Romani and G. Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. *Information Processing Letters*, 8(5):234-235, 1979.
- [3] D. Bini. Relations Between Exact and Approximate Bilinear Algorithms. Applications. *Calcolo* 17, 87–97, 1980
- [4] D. Coppersmith, S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation* 9, 3 (1990), 251–280
- [5] A. M. Davie, A. J. Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh* 143A, 2013, 351–370
- [6] C.-É. Drevet, M. N. Islam, É. Schost. Optimization techniques for small matrix multiplication. *Theoretical Computer Science*, Volume 412, Issue 22, (2011), 2219–2236
- [7] J. E. Hopcroft, L. R. Kerr. On minimizing the number of multiplications necessary for matrix multiplication. *SIAM Journal on Applied Mathematics*, Volume 20, Number 1, (1971), 30–35
- [8] J. E. Hopcroft, L. R. Kerr. Some techniques for proving certain simple programs optimal. *Proc. Tenth Ann. Symposium on Switching and Automata Theory*, 1969, 36–45
- [9] J. E. Hopcroft, J. Musinski. Duality applied to the complexity of matrix multiplications and other bilinear forms. *STOC '73 Proceedings of the fifth annual ACM symposium on Theory of computing*, (1973), 73–87, New York, NY, USA, ACM Press
- [10] M. N. Islam. Optimization Techniques For Matrix Multiplication. *M.Sc. thesis, University of Western Ontario*, 2009
- [11] J. D. Laderman. A Non-Commutative Algorithm for Multiplying 3×3 Matrices Using 23 Multiplications. *Bulletin of the American Mathematical Society*, Volume 82, Number 1, (1976), 126–128
- [12] F. Le Gall. Powers of tensors and fast matrix multiplication. *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC 2014)*, (2014), 296–303
- [13] O. M. Makarov. An algorithm for multiplication of 3×3 matrices. *Zh. Vychisl. Mat. Mat. Fiz.*, 26:2 (1986), 293–294
- [14] A. Sedoglavic. Fast matrix multiplication database. <https://fmm.univ-lille.fr>
- [15] A. Smirnov. An Approximate Bilinear Algorithm for Multiplying 5×5 Matrices of Length 90. <https://www.researchgate.net/publication/329800232>, 2018
- [16] A. V. Smirnov. The bilinear complexity and practical algorithms for matrix multiplication. *Zh. Vychisl. Mat. Mat. Fiz.*, Volume 53, Number 12, (2013), 1970–1984
- [17] A. J. Stothers. On the Complexity of Matrix Multiplication. *PhD thesis, University of Edinburgh*, 2010
- [18] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik* 13 (1969), 354–356
- [19] A. Waksman. On Winograds algorithm for inner products. In *IEEE Transactions on Computers*, C-19(1970), 360–361.

- [20] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th ACM Symposium on Theory of Computing*, 887–898, 2012
- [21] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. Version available at <http://theory.stanford.edu/~virgi/matrixmult-f.pdf>, retrieved on August 03, 2018
- [22] S. Winograd. On multiplication of 2×2 matrices. *Linear Algebra and its Applications*, Volume 4, Issue 4, (1971), 381–388