2015

Matrix Multiplication, Trilinear Decompositions, APA Algorithms, and Summation

Victor Pan
Lehman College and Graduate Center of CUNY

Matrix Multiplication, Trilinear Decompositions, APA Algorithms, and Summation

Victor Y. Pan

Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
and
Ph.D. Programs in Mathematics and Computer Science
The Graduate Center of the City University of New York
New York, NY 10036 USA
victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

Abstract

Matrix multiplication (hereafter we use the acronym MM) is among the most fundamental operations of modern computations. The efficiency of its performance depends on various factors, in particular vectorization, data movement and arithmetic complexity of the computations, but here we focus just on the study of the arithmetic cost and the impact of this study on other areas of modern computing. In the early 1970s it was expected that the straightforward cubic time algorithm for MM will soon be accelerated to enable MM in nearly quadratic arithmetic time, with some far fetched implications. While pursuing this goal the mainstream research had its focus on the decrease of the classical exponent 3 of the complexity of MM towards its lower bound 2, disregarding the growth of the input size required to support this decrease. Eventually, surprising combinations of novel ideas and sophisticated techniques enabled the decrease of the exponent to its benchmark value of about 2.38, but the supporting MM algorithms improved the straightforward one only for the inputs of immense sizes. Meanwhile, the communication complexity, rather than the arithmetic complexity, has become the bottleneck of computations in linear algebra. This development may seem to undermine the value of the past and future research aimed at the decrease of the arithmetic cost of MM, but we feel that the study should be reassessed rather than closed and forgotten. We review the old and new work in this area in the present day context, recall some major techniques introduced in the study of MM, discuss their impact on the modern theory and practice of computations for MM and beyond MM, and link one of these techniques to some simple algorithms for inner product and summation.

2000 Math. Subject Classification: 68Q25, 65F05, 15A06, 15A69, 01A60, 15-03

Key Words: Matrix multiplication, Computations, Complexity, Linear algebra, Tensor decomposition, Multilinear algebra, Inner product, Summation, Binary Segmentation

1 Introduction

Matrix multiplication (hereafter we keep using the acronym *MM*) is fundamentally important for computations in linear algebra and for the theory of computing. Efficient performance of MM depends on various factors, particularly on vectorization, data locality, and arithmetic cost (cf. [41, Chapter 1]). We review just the work on decreasing the arithmetic cost and comment on the impact of this work on modern computing beyond MM. Our review is a revision of [73] in the light of the information from the last 30 years.

The cubic arithmetic time $2n^3 - n^2$ of the straightforward algorithm for $MM(n)$, that is, for $n \times n$ MM, was commonly believed to be optimal until 1969, when it was decreased to $O(n^{2.8074})$ in [87], implying the decrease of the classical exponent 3 to 2.8074 also for numerous other venerated computational problems such as Boolean MM and the solution of a nonsingular linear system of n equations. The worldwide interest to MM has immediately exploded, and it was widely expected

that new efficient algorithms would soon perform MM and solve the related computational problems in nearly quadratic time. Even the exponent 2.8074, however, defied the attacks of all experts worldwide for almost 10 years, until 1978, and since then the mainstream research has been directed to the decrease of the exponents of MM allowing unrestricted input sizes. New surprising resources have been found, sophisticated techniques have been developed, and the exponents have eventually decreased to the current record level of about 2.38. The overhead constants hidden in the "O" notation of $O(n^{2.38})$ have been notoriously huge, however, and the supporting algorithms beat the straightforward one only for $n \times n$ input matrices of immense sizes. For n restricted to be "moderate", say, less than 1,000,000, the current record is just 2.7734 of [72], unbeaten since 1982, and the prospects for serious practical impact are even more bleak now. Here is a relevant citation from [6]: "The traditional metric for the efficiency of a numerical algorithm has been the number of arithmetic operations it performs. Technological trends have long been reducing the time to perform an arithmetic operation, so it is no longer the bottleneck in many algorithms; rather, communication, or moving data, is the bottleneck".

In the context of this development, we refocus our presentation versus [73]. We realize that many scientists are still curious whether MM and numerous related computational problems can be performed in nearly quadratic time, even for unrestricted input size. We cover briefly the effort motivated by this challenge and in Section 7 summarize the progress. Instead of various amazing sophisticated techniques proposed exclusively for the acceleration of MM of immense sizes, however, we cover the techniques that are efficient already for MM of moderate sizes or had interesting impacts on realistic computations beyond MM. The impacts have been limited to a few items, but played rather important role in the past and still have unexplored links to some apparently unrelated areas of computation, as in the case of our extension of an old MM technique to the computation of an inner product and summation (see Examples 8.1 and 8.2 in Section 8).

We organize our paper as follows. In Sections 2 and 4 we recall the two initial accelerations of MM, in 1969 and 1978, respectively, and comment on their impacts beyond MM. In Sections 3, 5, and 6 we cover the fundamental classes of bilinear, trilinear and the so called APA algorithms, respectively, and discuss their role in theoretical and practical acceleration of MM. In Section 7 we summarize the history of the subject and comment on its current state and prospects. In Section 8 we extend a fundamental MM technique to computing inner products and summation.

2 The Breakthrough in 1969: from the Exponent 3 to 2.8074

In addition to the acronym "MM" for "matrix multiplication", we write "MI" for "nonsingular matrix inversion", $MM(m,k,n)$ for the computation of the product $C = AB$ of two matrices A of the size $m \times k$ and B of the size $k \times n$, $MM(n)$ for $M(n,n,n)$, and $MI(n)$ for $n \times n$ MI. $A = (a_{i,j})_{i,j=1}^{m,n}$ denotes an $m \times n$ matrix with the entries $a_{i,j}$, for $i = 1, \dots, m$ and $j = 1, \dots, n$.

At first we recall the algorithm of [87], which performs $MM(n)$ for any n by using cn^{ω} arithmetic operations overall, for $\omega = \log_2(7) < 2.8074$ and a constant c independent of n, versus $2n^3 - n^2$ in the straightforward algorithm. The basic step of [87] is the computation of the product $C = AB$ of a pair of 2×2 matrices,

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix},\ B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix},\ C = AB = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

by using the following expressions, $p_1 = (a_{11}+a_{22})(b_{11}+b_{22})$, $p_2 = (a_{21}+a_{22})b_{11}$, $p_3 = a_{11}(b_{12}-b_{22})$, $p_4 = (a_{21}-a_{11})(b_{11}+b_{12})$, $p_5 = (a_{11}+a_{12})b_{22}$, $p_6 = a_{22}(b_{21}-b_{11})$, $p_7 = (a_{12}-a_{22})(b_{21}+b_{22})$, $c_{11} = p_1 + p_6 + p_7 - p_5$, $c_{12} = p_3 + p_5$, $c_{21} = p_2 + p_6$, $c_{22} = p_1 + p_3 + p_4 - p_2$.

The algorithm performs 7 scalar multiplications, instead of 8 of the straightforward algorithm, and this enables us to decrease the classical MM exponent 3 to 2.8074. Indeed, assume that the entries of the matrices A, B, and C above are the $2^{p-1} \times 2^{p-1}$ blocks of a 2×2 block matrix and apply the algorithm to this block matrix. Then the scalar multiplications turn into multiplications of the pairs of $2^{p-1} \times 2^{p-1}$ matrices. View them as the pairs of 2×2 block matrices with $2^{p-1} \times 2^{p-1}$ entries and apply the same algorithm again. Recursively apply the algorithm to the 2×2 block

matrices with the blocks of the sizes $2^{p-i} \times 2^{p-i}$, for $i = 1, 2, \dots, p$. Overall $MM(2^p)$ involves only 7^p scalar multiplications rather than the straightforward 2^{3p}. By embedding the $n \times n$ input matrices into $2^p \times 2^p$ matrices banded with zeros, perform $MM(n)$ by using 7^p multiplications in the case of any n such that $n \le 2^p < 2n$. Additions and subtractions of pairs of $n \times n$ matrices are relatively fast, each using only n^2 arithmetic operations, and we can extend the algorithm to performing $MM(n)$ for any n at the claimed arithmetic cost $cn^{\log_2(7)}$. With some additional care, one can yield the constant $c \approx 4.54$ and extend the complexity bounds $O(n^{\log_2(7)})$ to $MI(n)$, the solution of a nonsingular linear system of n equations, Boolean $MM(n)$, and a number of other computational problems well known in Computer Science, Linear Algebra, and Computer Algebra (see [87], [15], [1, Sections 6.3–6.6], [12, pages 49–51], and [11, Chapter 2], [28], [100], [**?**], [**?**], [47], [4], [81], [2]).

3 Bilinear Algorithms, Their Ranks and the MM Exponents

The above algorithm for $MM(2)$ belongs to the important class of bilinear algorithms. Such an algorithm for $MM(m, k, n)$ computes at first some linear forms $l_q(A)$ and $l'_q(B)$ in the entries of the input matrices $A = (a_{ij})_{i,j=1}^{m,k}$ and $B = (b_{jh})_{j,h=1}^{k,n}$ and then the entries $c_{ih} = \sum_j a_{ij} b_{jh}$ of the product $C = AB$ as the mn bilinear forms,

$$l_q(A) = \sum_{i,j=1}^{m,k} u_{ij}^{(q)} a_{ij},\ l'_q(B) = \sum_{j,h=1}^{k,n} v_{jh}^{(q)} b_{jh},\ q = 1, \dots, r,$$

$$c_{ih} = \sum_{q=1}^{r} w_{ih}^{(q)} l_q(A) l'_q(B), \quad i = 1, \dots, m;\ h = 1, \dots, n.$$

Here r is said to be the *rank of the algorithm*, $u_{ij}^{(q)}$, $v_{jh}^{(q)}$, and $w_{ih}^{(q)}$ are constants, for all i, j, h, and q, and the entries of the matrices A and B are variables or block matrices. Suppose $m = k = n$, assume that the entries are block matrices, and proceed as in the previous section, that is, apply recursively this bilinear algorithm of rank r to block matrices. This produces bilinear algorithms of rank $r^p = cn^\omega$ for $MM(n^p)$, $p = 2, 3, \dots$, and one can extend the algorithms to perform $MM(K)$ for any K by using cK^ω arithmetic operations for $\omega = \log_n(r)$ and a constant c independent of K.

The straightforward algorithm for $MM(m, k, n)$ is bilinear of rank mkn. The algorithm of [87] for $MM(2)$, reproduced in our Section 2, is bilinear of rank 7, and [66, Theorem 3] as well as [27] provide a formula for all bilinear algorithms of rank 7 for $MM(2)$. One of them, by Winograd (cf. [38], [12, pages 45–46], [1, Exercise 6.5], or [31]) performs $MM(2)$ by using 7 scalar multiplications and only 15 scalar additions and subtractions, instead of 18 in [87], and this is optimal for $MM(2)$ algorithms that use 7 scalar multiplications [77]. The Winograd algorithm still implies the same MM exponent $\log_2(7) < 2.8074$, but its careful implementation in [38] enables the decrease of the overhead constant from 4.54 to 3.95.

One can define bilinear algorithms for any *bilinear problem*, that is, for the computation of any set of bilinear forms, e.g., the product of two complex numbers $(a_1 + \mathbf{i}a_2)(b_1 + \mathbf{i}b_2) = (a_1b_1 - a_2b_2) + \mathbf{i}(a_1b_2 + a_2b_1)$, $\mathbf{i} = \sqrt{-1}$. The straightforward bilinear algorithm has rank 4, and here is a rank-3 bilinear algorithm, $l_1l'_1 = a_1b_1$, $l_2l'_2 = a_2b_2$, $l_3 = (a_1 + a_2)(b_1 + b_2)$, $a_1b_1 - a_2b_2 = l_1l'_1$, $a_1b_2 + a_2b_1 = l_0l'_0 - l_1l'_1$. See [96], [35], [36], [66], [14], [45], [88], [78], on the early study of bilinear algorithms and see a concise exposition in [12]. The book [97] shows efficient bilinear algorithms for the latter task and for the highly important computation of the product of 2 polynomials (that is, the convolution of their coefficient vectors), with further applications to the design of FIR-filters.

The minimal rank of all bilinear algorithms for a fixed bilinear problem such as $MM(m, k, n)$ is called the *rank of the problem*. It can be bounded in terms of the minimal arithmetic cost of the solution of the problem and vice versa [66, Theorem 1], [12], but not all interesting algorithms for bilinear problems are bilinear. The algorithm of [95] performs $MM(n)$ for any even n by using $(0.5n + 1)n^2$ scalar multiplications and $(1.5n^2 + 2n - 2)n$ additions and subtractions and has been refined slightly in [93]. These algorithms save about 50% of n^3 scalar multiplications, but they are

not bilinear. They use commutativity of the products of the entries of the input matrices and thus cannot be applied to block matrices, cannot be used recursively, and have made no impact on the exponent of MM. The algorithms, however, are of interest for MM of small sizes [32].

4 Bilinear Dead End and Trilinear Exit

The paper [87] has prompted the experts around the globe to compete for the next breakthrough towards the goal of performing MM in quadratic arithmetic time, but even the exponent 2.8074 defied all attacks for almost a decade, from 1969 to 1978. The research in these years was at first directed towards devising bilinear algorithms of rank 6 for $MM(2)$ and of rank 21 for $MM(3)$. Such algorithms would imply the decrease of the exponent 2.8074, but rank 7 turned out to be the sharp lower bound for $MM(2)$ (see [44]), while we still do not know whether the rank of $MM(3)$ exceeds 20 or not. We refer to [44], [66, Theorem 1], and [14] for the early lower bounds on the arithmetic complexity and on the rank of the problem $MM(m,k,n)$ for any triple of m, k, and n as well as on these bounds in the case of the specific problems $MM(2,2,n)$, $MM(2,3,3)$, $MM(2,3,3)$, $MM(2,3,4)$, and $MM(2,4,4)$. The paper [58] presents the current record lower bounds on the rank of $MM(n)$ for all n, while the papers [32] and [85] cover various lower and upper bounds on the arithmetic complexity and the rank of MM of smaller sizes.

The stalemate in 1969–1978 ended when the paper [67] presented a bilinear algorithm of rank 143,640 for $MM(70)$. This implied the exponent $\omega = \log_{70}(143,640) < 2.7962$ for $MM(n)$, $MI(n)$, Boolean $MM(n)$, and various other computational problems. The algorithm of [67] has extended an algorithm of the paper [66] of 1972, published in Russian[1] and translated into English only in 2014 in [76]. At first the paper reduced the acceleration of MM to the decomposition of a certain trilinear form into the sum of fewer trilinear products, which was a simple but novel step. Then the paper introduced a nontrivial technique of *trilinear aggregation* for generating such decompositions.

The technique has been the basis of the algorithm of [67] and has become indispensable for almost all subsequent decreases of the MM exponent, but the paper [66] was also a historical landmark in the study of multilinear and tensor decompositions. Such decompositions have been introduced by Hitchcock in 1927, but received scant attention until a half of a dozen of papers appeared in 1963–70 in the psychometrics literature. The paper [66] of 1972 was the next significant step: it presented the earliest known application of nontrivial multilinear and tensor decompositions to fundamental matrix computations. The paper has rarely been cited in the Western literature on MM and never in the works on multilinear and tensor decompositions, even though by now such decompositions have become a popular tool in linear and multilinear algebra and have a wide range of important applications to modern computing (see [91], [53], [62], [41], and the bibliography therein).

5 Trilinear Decompositions and Trilinear Aggregation

As we said, the paper [66] proposed and exploited trilinear representation in order to accelerate MM. Next we outline this approach and link it to some important computations beyond MM.

At first assume a bilinear algorithm of rank r for a matrix product $C = AB$ of an $m \times k$ matrix A by a $k \times n$ matrix B. Multiply its equations $\sum_j a_{i,j}b_{jh} = \sum_{s=1}^{r} w_{ih}^{(s)} l_s(A) l'_s(B)$ by new variables d_{hi}, sum the products in i and h, and arrive at a *trilinear decomposition* of rank r for $trace(ABD) = \sum_{i,j,h} a_{i,j}b_{jh}d_{hi}$ where $D = (d_{hi})_{h,i}^{n,m}$ is an auxiliary $n \times m$ matrix and $trace(M)$ denotes the trace of a matrix M. (This is equivalent to the decomposition of rank r for the tensor associated to the trilinear form $trace(ABD)$, that is, to its decomposition into the sum of r tensors of rank 1.) For example, here are the trilinear extensions of the bilinear algorithms of the previous section, that is, a trilinear decomposition of rank 3 for multiplication of two complex numbers,

$$a_1b_1d_1 - a_2b_2d_1 + a_1b_2d_2 + a_2b_1d_2 = a_1b_1(d_1 - d_2) - a_2b_2(d_1 + d_2) + (a_1 + a_2)(b_1 + b_2)d_2,$$

[1] Until 1976 the author lived in the Soviet Union. From 1964 to 1976 he has been working in Economics to make his living and has written the papers [65] and [66] in his spare time.

and a trilinear decomposition of rank 7 for $MM(2)$,

$$\sum_{i,j,h=1}^{2} a_{ij}b_{jh}d_{hi} = \sum_{s=1}^{7} l_s l'_s l''_s,\ \ l_1l'_1l''_1 = (a_{11}+a_{22})(b_{11}+b_{22})(d_{11}+d_{22}),$$

$l_2l'_2l''_2 = (a_{21}+a_{22})b_{11}(d_{12}-d_{22})$, $l_3l'_3l''_3 = a_{11}(b_{12}-b_{22})(d_{21}+d_{22})$, $l_4l'_4l''_4 = (a_{21}-a_{11})(b_{11}+b_{12})d_{22}$,
$l_5l'_5l''_5 = (a_{11}+a_{12})b_{22}(d_{21}-d_{11})$, $l_6l'_6l''_6 = a_{11}(b_{21}-b_{11})(d_{11}+d_{12})$, $l_7l'_7l''_7 = (a_{12}+a_{22})(b_{21}+b_{22})d_{11}$.

Conversely, interpret both sides of a decomposition of the trilinear form $trace(ABD)$ as linear forms in the variables d_{ih}, equate the coefficients of these variables on both sides of the decomposition, and come back to the original bilinear algorithm for the matrix product $C = AB$. By formalizing the above observations, [66, Theorem 2] states the equivalence of the design of a bilinear algorithm of rank r for $MM(m,k,n)$ to a trilinear decomposition of rank r for the associated trilinear form and its tensor.

By equating the coefficients of all variables a_{ij} or all variables b_{jh} on both sides of the trilinear decomposition, we obtain 2 other dual bilinear algorithms of the same rank for the problems $M(k,n,m)$ and $M(n,m,k)$. By interchanging the subscripts of the variables, we arrive at the dual bilinear algorithms of the same rank for the problems $MM(m,n,k)$, $MM(k,m,n)$, and $MM(n,k,m)$ as well (cf. [66, part 5 of Theorem 1], [14], [45], [78], [97]). The latter extension from triples to 6-tuples of algorithms uses the double subscripts for matrix entries and is peculiar to MM, but the triples of bilinear algorithms can be generated from their common trilinear representation for any bilinear computational problems, and the book [97] applies this technique in order to devise new efficient bilinear algorithms for FIR-filters and multiplication of complex numbers and polynomials.

For another demonstration of this duality technique, let us deduce part 1 of [66, Theorem 1], which states that we can perform $MM(K)$ by using cK^{ω} arithmetic operations for the exponent $\omega = 3\log_{mkn}(r)$ and a constant c independent of K provided that we are given a bilinear or trilinear algorithm of rank r for rectangular $MM(m,k,n)$ for any specific dimensions m, k, and n. In Section 3 we have already deduced this result in the case where $m = k = n$. Now successively apply the 3 dual bilinear algorithms of rank r to block $MM(m,k,n)$, block $MM(n,m,k)$, and $MM(k,n,m)$ and arrive at a bilinear algorithm of rank r^3 for $MM((mkn)^3)$. Then the claim follows from the result of Section 3 for n replaced by $(mkn)^3$.

Motivated by the basic fact of its Theorem 2 about the equivalence of bilinear and trilinear decompositions, the paper [66] has introduced by an example a nontrivial technique of trilinear aggregation for devising trilinear decompositions of small rank for $trace(ABD)$. Here is a similar example. At first assume that we seek two independent matrix products AB and UV and associate the pair of disjoint trilinear forms, $trace(ABD+UVW) = \sum_{i,j,h=1}^{m,k,n}(a_{ij}b_{jh}d_{hi} + u_{jh}v_{hi}w_{ij})$ to this task of *Disjoint MM*. Sum the trilinear aggregates $S = \sum_{i,j,h=1}^{m,k,n}(a_{ij}+u_{jh})(b_{jh}+v_{hi})(d_{hi}+w_{ij})$ and then subtract the correction terms $T_1 = \sum_{i,j=1}^{m,k} a_{ij}q_{ij}w_{ij}$, $T_2 = \sum_{j,h=1}^{k,n} b_{jh}u_{jh}r_{jh}$, and $T_3 = \sum_{h,i=1}^{n,m} p_{ih}v_{hi}d_{hi}$, where $p_{ih} = \sum_{j=1}^{k}(a_{ij}+u_{jh})$, $q_{ij} = \sum_{h=1}^{n}(b_{jh}+v_{hi})$, and $r_{jh} = \sum_{i=1}^{m}(d_{hi}+w_{ij})$. The resulting equation $trace(ABD+UVW) = S - T_1 - T_2 - T_3$ defines a trilinear decomposition of rank $mkn + mk + kn + nm$ (rather than the straightforward $2mkn$). Every aggregate is the products of 3 distinct multiplicands, each being the sum of 2 entries of the same column in the first Generating Table below. The correction terms are made up of the cross-products of the triples of the entries from 3 distinct columns of the table such that the 3 entries do not lie in the same row.

The technique of trilinear aggregation is naturally linked to Disjoint MM (see [73], [74, Section 5], [75, Section 2], and [54] on this link), and one can readily devise fast Disjoint MM of reasonable sizes by using this link. The above construction of Disjoint MM, a similar one below, and the ones of [54], however, have been quite readily extended to $MM(n)$. In particular, by playing with odd and even subscripts of the matrix entries, the paper [66] has extended the above decomposition of $trace(ABD+UVW)$ to a bilinear algorithm of rank $0.5n^3+3n^2$ for $MM(n)$ and any even n. This implied the MM exponent $\log_n(0.5n^3+3n^2)$, which is less than 2.85 for $n = 34$.

By using the second and the third Generating Tables below, we can decrease this exponent further. Indeed sum the mkn aggregates $(a_{ij}+u_{jh}+x_{hi})(b_{jh}+v_{hi}+y_{ij})(d_{hi}+w_{ij}+z_{jh})$ of the second Generating Table, subtract order of n^2 correction terms, and obtain a decomposition of rank $n^3+O(n^2)$ for $trace(ABC+UVW+XYZ)$, versus the straightforward $3n^3$. The trace represents 3 disjoint problems of $MM(n)$, that is, the computation of the 3 independent matrix products AB, UV, and XY, and we obtain a bilinear algorithm of rank $n^3+O(n^2)$ for this bilinear task.

Elaboration upon this idea in [67] has resulted in a trilinear decomposition and bilinear algorithms of rank $(n^3 - 4n)/3 + 6n^2$ for $MM(n)$, $n = 2p$, and any positive integer p. Substitute $n = 70$ and obtain the MM exponent 2.7962, cited in the previous section.

GENERATING TABLES:

a_{ij}	b_{jh}	d_{hi}
u_{jh}	v_{hi}	w_{ij}

a_{ij}	b_{jh}	d_{hi}
u_{jh}	v_{hi}	w_{ij}
x_{hi}	y_{ij}	z_{jh}

a_{ij}	b_{jh}	$\lambda^2 d_{hi}$
λu_{jh}	λv_{hi}	w_{ij}

6 APA Algorithms and Further Acceleration of MM

The third Generating Table helps us to demonstrate the technique of *Any Precision Approximation* introduced in [8] and [7] (hereafter we use the acronym *APA*). Its combination with trilinear aggregation and Disjoint MM has enabled further decrease of the MM exponents. Moreover, our Section 8 and [75, Section 40] link the APA technique to some fundamental computations beyond MM.

For a sample APA algorithm, generate the sum of mkn aggregates $S = \lambda^{-1} \sum_{i,j,h=1}^{m,k,n} (a_{ij} + \lambda u_{jh})(b_{jh} + \lambda v_{hi})(\lambda^2 d_{hi} + w_{ij})$ from the third Generating Table above. For $\lambda = 1$ this table turns into the first Generating Table, and then trilinear aggregation of the previous section enables a decomposition of rank $mkn + mk + kn + mn$ for $trace(ABD + UVW)$. Suppose, however, that $\lambda \to 0$ and obtain a trilinear decomposition $trace(ABD + UVW) = S - T_1 - T_2 + O(\lambda)$, where $T_1 = \sum_{i,j=1}^{m,k} a_{ij} q_{ij} w_{ij}$ and $T_2 = \sum_{j,h=1}^{k,n} b_{jh} u_{jh} r_{jh}$, for $q_{ij} = \sum_{h=1}^{n} (b_{jh} + \lambda v_{hi})$ and $r_{jh} = \sum_{i=1}^{m} (\lambda^2 d_{hi} + w_{ij})$. If we ignore the terms of order λ, then the rank of the decomposition turns into its *λ-rank* or *border rank* and decreases to $mkn + mk + kn$.

By equating the coefficients of the variables d_{hi} and w_{ij} in the trilinear APA decomposition above, we arrive at the bilinear problem of the evaluation of two disjoint matrix products AB and UV. The mkn trilinear aggregates turn into the mkn bilinear products $(a_{ij} + \lambda u_{jh})(b_{jh} + \lambda v_{hi})$ for all i, j, and h. Clearly, $a_{ij} + \lambda u_{jh} \to a_{ij}$ and $b_{jh} + \lambda v_{hi} \to b_{jh}$ as $\lambda \to 0$, but we must keep the terms λu_{jh} and λv_{hi} in the aggregates in order to compute the matrix product UV. This can force us to double the precision of the representation of the multiplicands $a_{ij} + \lambda u_{jh}$ and $b_{jh} + \lambda v_{hi}$ compared to the precision of the representation of the entries a_{ij}, u_{jh}, b_{jh}, and v_{hi}. E.g., suppose that $\lambda = 2^{-s}$ for a sufficiently large integer s and that a_{ij}, u_{jh} b_{jh}, and v_{hi} are s-bit integers in the range $[0, 2^s)$. Then $2s$ bits are required to represent each of the multiplicands $a_{ij} + \lambda u_{jh}$ and $b_{jh} + \lambda v_{hi}$.

One would prefer to avoid doubling the precision if s exceeds the half-length of the computer word, but in [7] Bini proved that we can ignore this problem if our goal is just the decrease of the exponents of MM. Indeed, multiply the above trilinear decomposition of $trace(ABD+UVW)$ by the variable λ and arrive at a trilinear decomposition whose coefficients are polynomials in λ of degree $d = 2$. In particular $trace(ABD + UVW)$ is the coefficient of λ in this decomposition, which we can compute by means of interpolation.

For decreasing the exponent of MM, this observation is not sufficient yet, because the interpolation stage increases the rank by a factor of $(d + 1)^2$, that is, by a factor of 9 for $d = 2$. In the above case the resulting rank $9(mkn + mk + kn)$ exceeds the rank $2mkn$ of the straightforward algorithm for $trace(ABD + UVW)$. After sufficiently many recursive applications of the algorithm, however, the comparison is reversed because in every recursive step the degree of a decomposition in λ is only doubled, whereas the matrix size is squared, that is, the degree grows logarithmically in the matrix size. Ultimately the impact of the interpolation factor on the MM exponent becomes immaterial. Hence the MM exponent $3 \log_{mkn} r$ is defined even by an APA trilinear decomposition of $trace(ABD)$ having border rank r if $trace(ABD)$ represents $MM(m, k, n)$ for fixed m, k, and n.

The APA decomposition above for $trace(ABD + UVW)$ is associated with Disjoint MM rather than MM, but Schönhage in [82] proved that the exponents of MM can be obtained readily from the decompositions for Disjoint MM as well. In particular the above APA decomposition of $trace(ABD + UVW)$ implies the MM exponent $\omega = 3 \log_{mkn}(0.5(mkn + mk + kn))$ for any triple of m, k, and n. Substitute $m = n = 7$ and $k = 1$ and obtain the MM exponent $\omega = 3 \log_{49} 31.5 < 2.66$. Further refinement of this construction in [82] yielded the exponent $\omega < 2.548$, and [71] decreased this record to $\omega < 2.522$ by combining APA algorithms and trilinear aggregation for disjoint MM represented by $trace(ABC + UVW + XYZ)$.

By continuing this line of research, the next 2 celebrated papers [89] and [24] decreased the exponents to the record of about 2.38, which was further decreased by 0.002 in 2010 and 0.001 in 2012–2014, respectively. Namely, due to the results in [66, part 1 of Theorem 1 and Theorem 2], [7], [82], [89], and [24] it became possible and challenging to deduce small exponents of MM from more and more lenient basic bilinear or trilinear decompositions of small rank for MM and Disjoint MM of small sizes, and since 1986 for some small bilinear problems similar to Disjoint MM. All these results have been built on the top of the techniques of the preceding papers and have been increasing in sophistication. Accordingly they required performing longer recursive processes of nested block MMs, and the input size had to be blown up to accommodate these processes. This deficiency has been avoided by trilinear aggregation, but by none of the subsequent techniques.

7 A Summary of the Research on Fast MM

Let us summarize. Strassen's seminal paper [87] of 1969 has motivated extensive effort worldwide with the goal of decreasing the exponent of MM towards its information lower bound 2. In spite of the initial optimism, almost a decade of stalemate followed. Then, since 1978, application of trilinear aggregation and, since 1979, its combination with the techniques of APA algorithms and Disjoint MM enabled further decrease of the MM exponents. Strassen's exponent 2.8074 of 1969 has been beaten a number of times in 1978–1981, ending with 2.496 of [23]. Two new decreases in 1986 produced the benchmark record 2.376 of [24], decreased by tiny margins, to 2.374 in [86] and [26] and 2.373 in [92] and [56]. (The title of [92] is deceptive a little. The paper has decreased the fresh exponent of [86] and [26], but not the decades-old exponent of [24].) Almost all decreases of the MM exponents relied on amazing novel techniques built on the top of each other. According to [24, page 255], all of them employed trilinear aggregation and since 1979 combined it with APA algorithms.

Already in [82] Schönhage pointed out, however, that all new exponents of MM were just "of theoretical interest" because they were valid only for the inputs "beyond any practical size", and so, according to [82], "Pan's estimates of 1978 for moderate" input sizes were "still unbeaten". Actually the exponent 2.7962 of 1978 for $MM(n)$ (restricted, say, to $n \leq 1,000,000$) has successively been decreased in [68], [69], [71], and [72] (cf. also [73]), although by small margins. As of November 2014, the exponent 2.7734 of [72] is still the record low for $MM(n)$ with $n \leq 1,000,000$.

Tables 7.1 – 7.3 and Figures 1 and 2 show the dynamics of the exponents of $MM(n)$ since 1969, under no restriction on the dimension n and for $n \leq 1,000,000$, respectively. The tables link each exponent to its recorded publication in a journal, a conference proceedings, or as a research report. Accordingly, Figures 1 and 2 display the chronological process and reflect the competition for the decrease of the MM exponents, particularly intensive in 1979–1981 and 1986. The record exponents of MM have been updated 4 times during the single year of 1979. At first the MM exponent 2.7801 appeared in February in a Research Report (see [69]). 2.7799 appeared as an APA MM exponent in [8] in March and then as an MM exponent in [7]. The next MM exponent 2.548 of [82] was followed by the MM exponent 2.522 of [71], both published in the book of abstracts of the conference on the Computational Complexity in Oberwolfach, West Germany, organized by Schnorr, Schönhage and Strassen (cf. [73, page 199] and [82]). The exponent 2.496 of [23] was reported in October 1981 at the IEEE FOCS'81, but in Table 7.1 and Figure 1 we place it after the exponent 2.517 of the paper [80] of 1982, which was submitted in March 1980. The Research Report version of the paper [24] appeared in August of 1986, but in Table 7.1 and Figure 1 we place [24] after the paper [89], published in October of 1986 in the Proceedings of the IEEE FOCS, because the paper [89] has been submitted to FOCS'86 in the Spring of 1986 and has been widely circulated afterwards. One could complete the historical account of Tables 7.1 and 7.2 and Figure 1 by including our exponent 2.7804 (announced in the Fall of 1978 but superseded in February 1979 in [69]) and the value 2.5218007, which decreased our exponent 2.5218128 of 1979 and appeared at the end of the final version of [82] in 1981, that is, before the publication, but after the submission of the exponent 2.517 of [80]. Table 7.2 shows the decrease of the exponents in 1986 – 2014.

We refer the reader to [21], [57], [22], [46], [50], [55], and the references therein for similar progress in asymptotic acceleration of rectangular MM and its theoretical implications.

Table 7.1: $MM(n)$ exponents for unrestricted n.

2.8074	2.7962	2.7801	2.7799	2.548	2.522	2.517	2.496	2.479	2.376	2.374	2.373
[87]	[67]	[69]	[8], [7]	[82]	[71]	[80]	[23]	[89]	[24]	[86], [26]	[92], [56]
1969	1978	1979	1979	1979	1979	1980	1981	1986	1986	2010	2012

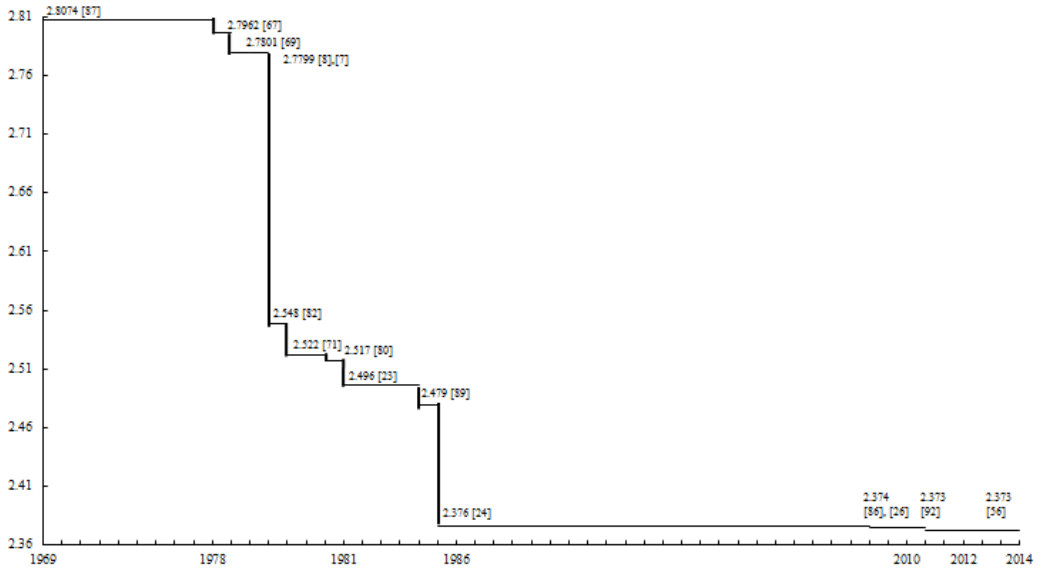


Figure 1: $MM(n)$ exponents for unrestricted n.

The decrease of the MM exponents displayed in Tables 7.1 and 7.2 and Figure 1 has been a major issue in Computer Science since 1969 and attracted a lot of attention worldwide. Only a very small part of the proposed algorithms, however, and nothing produced after 1971, have been used for practical MM so far. It should be clarified right away that, in spite of some initial fears and persistent reservations about the early algorithms by Strassen and Winograd, all fast MM algorithms can be implemented with no serious numerical stability problems (see [9], [30], [29]). Still the cited comments from Schönhage's paper [82] apply to all the acclaimed MM algorithms, supporting the record exponents since 1979.

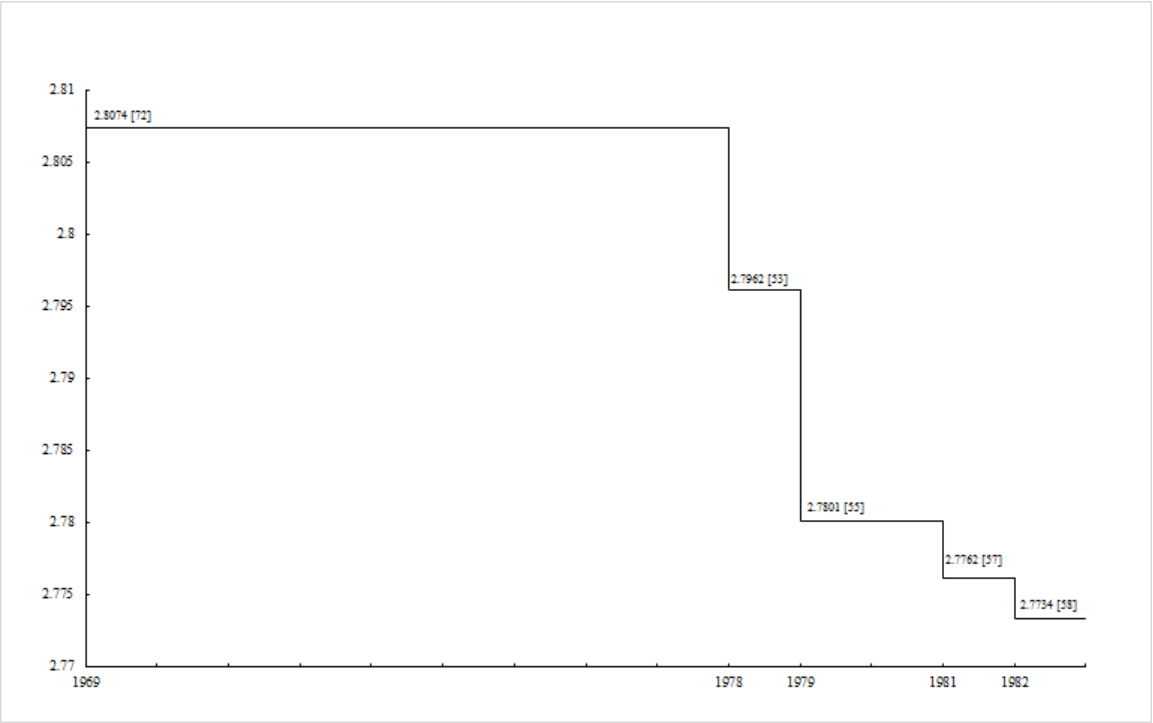
Table 7.3 and Figure 2 display the chronology of decreasing the record exponents of $MM(n)$ for $n \leq 1,000,000$. The associated overhead constants are small because the supporting algorithms avoid recursive application of nested block MM and rely just on the trilinear aggregation techniques of [66], [67], and [71]. Some of these algorithms have also been refined in [73], [54] and then implemented in [48] and [49].

The MM algorithms used in practice (cf. [5], [43], [31], [33], [34], [13], [25], the references therein, and the detailed account in [41, Chapter 1]) either rely just on the straightforward algorithm or employ the Strassen and Winograd algorithms supporting the MM exponent $\log_2(7) \approx 2.8074$ and in some cases combined with the algorithms of [95] and [93], applied for matrices of smaller sizes. These implementations make up a relatively small, but valuable part of the present day MM software. Kaporin's implementations in [48] and [49] promise further benefits, in particular they use small memory space, and in part (ii) of our comments below we point out another important resource for

Table 7.2: $MM(n)$ exponents for unrestricted n in 1986–2014.

2.3754770	2.3736898	2.3729269	2.3728639
[24]	[86], [26]	[92]	[56]
1986	2010	2012^2	2014

Figure 2: $MM(n)$ exponents for $n \leq 1,000,000$.



further progress.

Even though the impact of the study of fast MM on computational practice has been limited and in spite of the cited decrease of the importance of the arithmetic complexity versus communication complexity of the computations in linear algebra [6], we feel that further study of the arithmetic complexity of MM can still be interesting if one reassesses its goals.

(i) It is still possible that new breakthrough will enable MM in quadratic arithmetic time up to a logarithmic or polylogarithmic factor. For a while the researchers expected to see such a breakthrough from the group theoretical approach to MM in [18] and [20], but the papers [2], [19], and [3] show that this dream is not going to come true and that there are similar limitations of the recent progress in [89], [24], [86], [26], [92], and [56]. We may still hope for breakthroughs based on other surprising ideas and impressive advanced techniques, but should not forget the cited caveat from [82] about the toll for working with MM of unrestricted sizes.

(ii) The combination of human and computer research towards devising efficient MM algorithms has good chances for success in the future. Such a combination has produced the seminal algorithm of [8], according to [79], and (combined with some relatively simple optimization technique) has recently enabled the acceleration of the known algorithms for some rectangular MM in the cases of practical interest, namely for $MM(m,k,n)$ where $\max\{m,k,n\} \leq 6$ and $\min\{m,k,n\} \leq 3$ (see [85]). One can try to combine the optimization technique of [85] with trilinear aggregation and computerized search in order to accelerate Disjoint rectangular MM of small sizes. The study in [85] has not determined the rank of $M(3,3,3)$ and has not beaten the exponent 2.7734 of [72], but

Table 7.3: $MM(n)$ exponents for $n \leq 1,000,000$.

2.8074	2.7962	2.7801	2.7762	2.7734
[87]	[67]	[69]	[71]	[72]
1969	1978	1979	1981	1982

produced algorithms that run faster than the Strassen, Winograd, and all other known algorithms for some small rectangular MM problems. This can already serve as a basis for some improvement of the present day MM software, but some more advanced optimization techniques supported by more powerful computers may lead to new efficient algorithms for rectangular MM and Disjoint MM of small sizes, to decreasing the exponent 2.7734 of $MM(n)$ for $n \leq 1,000,000$, unbeaten since 1982, and to revealing hidden but powerful MM techniques.

(iii) The research on MM had valuable by-products for some related subject areas. In addition to the impact of MM on the theory of computing, which we pointed out at the end of Section 2, let us recall that MM has served as a basic problem in the field of algebraic computations, important for both theory and practice of computing (cf. [1], [12], [11], [16], [52], [40]). The origin of the field can be traced back to [64], [59], [90], and [65], but the study of bilinear and trilinear algorithms for MM in the papers [94], [66], [14], [45], [88], [78], [77], [67], and [73] as well as APA algorithms in the papers [8] and [7] have been important steps in this field. The duality technique for generating new efficient bilinear algorithms, with applications shown in [97], exemplifies other valuable by-products of the MM study. Besides, as we said earlier, the paper [66] was pioneering in the study of tensor decompositions. We hope to see new impacts of the MM research on the theory and practice of computing in the future. In the next section we extend the study in [7] to some apparently unrelated algorithms for inner products and summation, having further link to tensor decomposition.

8 APA Approach and a Summation Algorithm

Recall the APA algorithm of Section 6, assume that the entries a_{ij}, b_{jh}, u_{jh}, and v_{hi} are integers in the range $[0, 2^d)$, and choose $\lambda = 2^d$. Then the product $(a_{ij} + \lambda b_{jh})(u_{jh} + \lambda v_{hi})$ would fit the length L of the computer word if $L \geq 4d$. Moreover if the ratio L/d is large enough, we can perform the APA computations of Section 6 within the precision L. [75, Section 40] exploits such observations further to devise efficient algorithms for multiplication of vectors and matrices filled with bounded integers. Next we recall that technique and in Example 8.2 show its surprising extension.

Suppose that the coefficient vector of a polynomial $v(\lambda) = \sum_{i=0}^{n-1} v_i \lambda^i$ is filled with integers from the semi-open segment $[0, 2^d)$ of the real axis for a positive integer d. Represent this vector by the 2^d-ary integer $v(2^d) = \sum_{i=0}^{n-1} v_i 2^{di}$. Generally the interpolation to a polynomial of degree $n-1$ requires its evaluation at n knots, but in the above special case we only need the evaluation at the single knot 2^d. Now suppose that all coefficients v_i are integers from the semi-open segment $[q, r)$ for any pair of integers q and r, $q < r$. Then we can apply the above recipe to compute the shifted vector $\mathbf{u} = (u_i)_{i=0}^{n-1} = (v_i - q)_{i=0}^{n-1}$, having all its components in the semi-open segment $[0, s)$ for $s = r - q$. Finally we would recover the vector $\mathbf{v}$ from $\mathbf{u}$. By following [70] and [10], we call this technique *binary segmentation*. Its history can be traced back to [37], and one can even view it as an application of the Kronecker map, although having specific computational flavor.

Next we follow [75, Example 40.3, pages 196–197] to compute the inner product of two integer vectors, then extend the algorithm to summation, and finally list various other applications of binary segmentation.

Example 8.1. *(*The inner product of two integer vectors*, cf. [75, Example 40.3].) Assume two nonnegative integers g and h and two vectors $\mathbf{u} = (u_i)_{i=0}^{n-1}$ and $\mathbf{v} = (v_i)_{i=0}^{n-1}$ with nonnegative integer coordinates in two semi-open segments, namely, $[0, 2^g)$ for the coordinates of $\mathbf{u}$ and $[0, 2^h)$ for the coordinates of $\mathbf{v}$. The straightforward algorithm for the inner product $\mathbf{u}^T\mathbf{v} = \sum_{i=0}^{n-1} u_i v_i$ at first computes the n products $u_i v_i$ for $i = 0, 1, \ldots, n-1$ and then sums them. This involves*

n multiplications and $n-1$ additions. Instead, however, we can just multiply a pair of bounded nonnegative integers, apply binary segmentation to the product, and output the desired inner product. Namely, introduce the two polynomials $u(x) = \sum_{i=0}^{n-1} u_i x^i$ and $v(x) = \sum_{i=0}^{n-1} v_i x^{n-1-i}$. Their product is the polynomial $q(x) = u(x)v(x) = \sum_{i=0}^{2n-2} q_i x^i$ with integer coefficients in the segment $[0, 2^k)$ for $k = g+h+\lceil \log_2 n \rceil$. The coefficient $q_{n-1} = \sum_{i=0}^{n-1} u_i v_i$ is precisely the inner product $\mathbf{u}^T\mathbf{v}$. Represent the polynomials $u(x)$ and $v(x)$ by their integer values $u(2^k)$ and $v(2^k)$ at the point 2^k. Clearly, they lie in the semi-open segments $r_u = [0, 2^{nk+g})$ and $r_v = [0, 2^{nk+h})$, respectively. Now compute the integer $q(2^k) = u(2^k)v(2^k)$, lying in the segment $[0, 2^{2nk+g+h})$, and recover the coefficient $q_{n-1} = \mathbf{u}^T\mathbf{v}$ by applying binary segmentation.

Remark 8.1. *We only seek the coefficient q_{n-1} of the median term $q_{n-1}x^{n-1}$ of the polynomial $u(x)v(x)$. This term lies in the segment $[2^{(n-1)k}, 2^{(n-1)k+g+h})$, and the next challenge is to optimize its computation. Is such a more narrow task substantially simpler than the multiplication of two integers lying in the segments r_u and r_v?*

Example 8.2. *(*Summation of bounded integers.*) For $\mathbf{u} = (1)_{i=0}^{n-1}$, $g = 0$, and $k = h + \lceil \log_2 n \rceil$ or $\mathbf{v} = (1)_{i=0}^{n-1}$, $h = 0$, and $k = g + \lceil \log_2 n \rceil$, the algorithm of Example 8.1 outputs the sum of n integers.*

Remark 8.2. *In the same way as for polynomial interpolation in the beginning of this section, we can relax the assumption of Examples 8.1 and 8.2 that the input integers are nonnegative. Moreover, the summation of integers can be extended to the fundamental problem of the summation of binary numbers truncated to a fixed precision.*

In Examples 8.1 and 8.2, multiplication of two long integers followed by binary segmentation replaces either $2n-1$ or n arithmetic operations, respectively. This increases the Boolean (bit-wise operation) cost by a factor depending on the Boolean cost of computing the product of 2 integers or, in view of Remark 8.1, of computing the median segment in the binary representation of the product. The increase is minor if we multiply integers in nearly linear Boolean time (see the supporting algorithms for such multiplication in [84], [1, Section 7.5], [39]), but grows if we multiply integers by applying the straightforward algorithm, which uses quadratic Boolean time. Nonetheless, in both cases one could still benefit from using Example 8.2 if the necessary bits of the output integer fits the computer word (i.e. the bits of the middle coefficient are not part of the overflow of the product), as long as the representation of the vector as an integer requires no additional cost. If the output integer does not fit the word length, we can apply the same algorithms to the subproblems of smaller sizes, e.g., we can apply the algorithms of Examples 8.1 and 8.2 to compute the inner products of some subvectors or partial sums of integers, respectively.

Other applications of binary segmentation include polynomial multiplication (that is, the computation of the convolution of vectors) [37], [83], some basic linear algebra computations [75, Examples 40.1–40.3], polynomial division [10], [83], computing polynomial GCD [17], and discrete Fourier transform [83]. Binary segmentation can be potentially efficient in computations with Boolean vectors and matrices. E.g., recall that Boolean MM is reduced to MM whose input and output entries are some bounded nonnegative integers (see [1, Proof of Theorem 6.9]). Quantized tensor decompositions is another promising application area (cf. [91], [60], [61], [51], [63], [42]).

Acknowlegements. My work has been supported by NSF Grant CCF–1116736 and PSC CUNY Award 67699-00 45. Furthermore I am grateful to A. Bostan, I.V. Oseledets and E.E. Tyrtyshnikov for their valuable pointers to some important recent works on MM and the bibliography on quantized tensor decompositions, respectively, to Franklin Lee, Tayfun Pay, and Liang Zhao for their helpful comments on potential implementation of the algorithms of Examples 8.1 and 8.2 as well as excellent assistance with creating Figures 1 and 2.

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Algorithms.* Addison-Wesley, Reading, MA, 1974.

[2] N. Alon, A. Shpilka, C. Umans. On Sunflowers and Matrix Multiplication. *Computational Complexity*, **22, 2**, 219–243, 2013.

[3] A. Ambainis, Y. Filmus, F. Le Gall, Fast Matrix Multiplication: Limitations of the Laser Method. Available at arXiv:1411.5414, November 21, 2014.

[4] R. R. Amossen, R. Pagh, Faster Join-projects and Sparse Matrix Multiplications. In *Proceedings of the 12th International Conference on Database Theory*, 121–126, 2009.

[5] D. Bayley, Extra High Speed Matrix Multiplication on the Cray-2. *SIAM J. on Scientific and Statistical Computing*, **9**, 603–607, 1988.

[6] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, O. Schwartz, Communication Lower Bounds and Optimal Algorithms for Numerical Linear Algebra. *Acta Numerica*, **23**, 1–155, 2014.

[7] D. Bini, Relations Between Exact and Approximate Bilinear Algorithms: Applications. *Calcolo*, **17**, 87–97, 1980.

[8] D. Bini, M. Capovani, G. Lotti, F. Romani, $O(n^{2.7799})$ Complexity for $n \times n$ Approximate Matrix Multiplication. *Information Processing Letters*, **8**, 234–235, March 1979.

[9] D. Bini and D. Lotti, Stability of Fast Algorithms for Matrix Multiplication. *Numerische Math.*, **36**, 63–72, 1980.

[10] D. Bini, V. Y. Pan, Polynomial Division and Its Computational Complexity. *J. Complexity*, **2**, 179–203, 1986.

[11] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*. Birkhäuser, Boston, 1994.

[12] A. Borodin, I. Munro, *The Computational Complexity of Algebraic and Numeric Problems.* American Elsevier, New York, 1975.

[13] B. Boyer, J.-G. Dumas, C. Pernet, W. Zhou, Memory Efficient Scheduling of Strassen-Winograd's Matrix Multiplication Algorithm. *Proc. Intern. Symposium on Symbolic and Algebraic Computation (ISSAC 2009)*, 55–62, ACM Press, New York, 2009.

[14] R.W. Brockett, D. Dobkin, On Optimal Evaluation of a Set of Bilinear Forms. *Proc. of the 5th Annual Symposium on the Theory of Computing (STOC 1973)*, 88–95, ACM Press, New York, 1973.

[15] J. R. Bunch and J. E. Hopcroft, Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, **28**, 231–236, 1974.

[16] P. Bürgisser, M. Clausen, M.A. Shokrollahi, *Algebraic Complexity Theory*. Springer Verlag, 1997

[17] B. W. Char, K. O. Geddes, G. H. Gonnet, QCDHUE: Heuristic Polynomial GCD Algorithm Based on Integer GCD Computation. *Proceedings of EUROSAM'84, Lecture Notes in Computer Science*, **174**, 285–296, Springer, New York, 1984.

[18] H. Cohn, C. Umans, A Group-theoretic Approach to Fast Matrix Multiplication. *Proceedings of the 44th Annual Symposium on Foundations of Computer Science*, (Cambridge, MA), 438–449, IEEE Computer Society, 2003.

[19] H. Cohn, C. Umans, Fast Matrix Multiplication Using Coherent Configurations. *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1074–1087, 2013.

[20] H. Cohn, R. Kleinberg, B. Szegedy, C. Umans, Group-theoretic Algorithms for Matrix Multiplication. *Proceedings of the 46th Annual Symposium on Foundations of Computer Science*, (Pittsburgh, PA), 379–388, IEEE Computer Society, 2005.

[21] D. Coppersmith, Rapid Multiplication of Rectangular Matrices. *SIAM Journal on Computing*, **11, 3**, 467–471, 1982.

[22] D. Coppersmith, Rectangular Matrix Multiplication Revisited. *Journal of Complexity*, **13, 1**, 42–49, 1997.

[23] D. Coppersmith, S. Winograd, On the Asymptotic Complexity of Matrix Multiplication. *SIAM J. on Computing*, **11, 3**, 472–492, 1982. Proc. version in *23rd FOCS*, 82–90, (Nashville, TN), IEEE Computer Society Press, 1981.

[24] D. Coppersmith, S. Winograd, Matrix Multiplicaton via Arithmetic Progressions. *J. of Symbolic Computations*, **9, 3**, 251–280, 1990. Proc. version in *19th ACM STOC 1987* (New York, NY), 1–6, ACM Press, New York, NY, 1987. Also Research Report RC 12104, *IBM T.J. Watson Research Center*, August 1986.

[25] P. D'Alberto, A. Nicolau, Adaptive Winograd's Matrix Multiplication. *ACM Transactions on Mathematical Software*, **36, 1**, paper 3, 2009.

[26] A. M. Davie, A. J. Stothers, Improved Bound for Complexity of Matrix Multiplication. *Proceedings of the Royal Society of Edinburgh*, **143A**, 351–370, 2013.

[27] H.F. de Groot, On Varieties of Optimal Algorithms for the Computation of Bilinear Mappings. *Theoretical Computer Science*, **7**, 1–24, 1978.

[28] C. Demetrescu, G. F. Italiano, Fully Dynamic Transitive Closure: Breaking Through the $o(n^2)$ Barrier. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*, 381–389, 2000.

[29] J. Demmel, I. Dumitriu, O. Holtz, Fast Linear Algebra Is Stable. *Numerische Mathematik*, **108, 1**, 59–91, 2007.

[30] J. Demmel, I. Dumitriu, O. Holtz, R. Kleinberg, Fast Linear Algebra Is Stable. *Numerische Mathematik*, **106, 2**, 199–224, 2007.

[31] C. C. Douglas, M. Heroux, G. Slishman, R. M. Smith, GEMMW: A Portable Level 3 BLAS Winograd Variant Of Strassen's Matrix-Matrix Multiply Algorithm. *J. of Computational Physics*, **110, 1**, 1–10, 1994.

[32] C.-E. Drevet, Md. N. Islam, E. Schost, Optimization Techniques for Small Matrix Multiplication. *Theoretical Computer Science*, **412 22**, 2219–2236, 2011.

[33] J.-G. Dumas, P. Giorgi, C. Pernet, FFPACK: Finite Field Linear Algebra Package. *Proc. Intern. Symposium on Symbolic and Algebraic Computation (ISSAC 2004)*, Jaime Gutierrez, editor, 119–126, ACM Press, New York (2004)

[34] J.-G. Dumas, P. Giorgi, C. Pernet, Dense Linear Algebra over Word-size Prime Fields: the FFLAS and FFPACK Packages. *ACM Trans. Math. Software*, **35, 3**, 1–42, 2008.

[35] C. M. Fiduccia, On Obtaining Upper Bound on the Complexity of Matrix Multiplication. In *Analytical Complexity of Computations* (edited by R.E. Miller, J. W. Thatcher, J. D. Bonlinger), pp. 31–40, Plenum Press, NY, 1972.

[36] C. M. Fiduccia, Polynomial Evaluation via the Division Algorithm: The Fast Fourier Transform Revisited. *Proc. 4th Annual ACM Symposium on Theory of Computing (STOC 1972)*, 88–93, ACM Press, New York, 1972.

[37] M.J. Fischer, M.S. Paterson, String-Matching and Other Products. *SIAM–AMS Proc.*, **7**, 113–125, 1974.

[38] P.C. Fischer, Further Schemes for Combining Matrix Algorithms. *Proceedings of the 2nd Colloquium on Automata, Languages and Programming*, 428–436 Springer-Verlag, London, UK, 1974.

[39] M. Fürer, Faster Integer Multiplication. *SIAM J. on Computing*, **39, 3**, 979–1005, 2009.

[40] J. von zur Gathen, J. Gerhard (2013). *Modern Computer Algebra*. Cambridge University Press, Cambridge, UK, third edition, 2013.

[41] G. H. Golub, C. F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 2013 (4th addition).

[42] L. Grasedyck, D. Kressner, C. Tobler, A Literature Survey of Low-rank Tensor Approximation Techniques. *GAMM-Mitteilungen*, **36**, **1**, 53–78, 2013.

[43] N. J. Higham, Exploiting Fast Matrix Multiplication within Level 3 BLAS. *ACM Trans. on Math. Software*, **16**, 352–368, 1990.

[44] J.E. Hopcroft, L.R. Kerr, Some Techniques for Proving Certain Simple Programs Optimal. *Proceedings of the Tenth Annual Symposium on Switching and Automata Theory*, 35–45, IEEE Computer Society Press, 1969.

[45] J.E. Hopcroft, J. Musinski, Duality Applied to Matrix Multiplication and Other Bilinear Forms. *SIAM Journal on Computing*, **2, 3**, 159–173, 1973.

[46] X. Huang, V. Y. Pan, Fast Rectangular Matrix Multiplication and Applications. *Journal of Complexity*, **14, 2**, 257–299, 1998. Proc. version in *Proc. Annual ACM International Symposium on Parallel Algebraic and Symbolic Computation (PASCO'97)*, 11–23, ACM Press, New York, 1997.

[47] H. Kaplan, M. Sharir, E. Verbin, Colored Intersection Searching via Sparse Rectangular Matrix Multiplication. In *Proceedings of the 22nd ACM Symposium on Computational Geometry*, 52–60, 2006.

[48] I. Kaporin, A Proctical Algorithm for Faster Matrix Multiplication. *Numerical Linear Algebra*, **6**, 687-700, 1999.

[49] I. Kaporin, The Aggregation and Cancellation Techniques as a Practical Tool for Faster Matrix Multiplication. *Theoretical Computer Science*, **315**, **2–3**, 469–510, 2004.

[50] S. Ke, B. Zeng, W. Han, V. Y. Pan, Fast Rectangular Matrix Multiplication and Some Applications. *Science in China, Series A: Mathematics*, **51**, 3, 389–406, 2008.

[51] B. N. Khoromskij, $O(d \log N)$ Quantics Approximation of $N - d$ Tensors in High-dimensional Numerical Modeling. *Constructive Approximation*, **34**, **2**, 257–280, 2011.

[52] D. E. Knuth, *The Art of Computer Programming: Volume 2, Seminumerical Algorithms*. Addison-Wesley, Reading, Massachusetts, 1969 (first edition), 1981 (second edition), 1998 (third edition).

[53] T.G. Kolda, B.W. Bader, Tensor Decompositions and Applications. *SIAM Review*, **51, 3**, 455–500, 2009.

[54] J. Laderman, V.Y. Pan, H.X. Sha, On Practical Algorithms for Accelerated Matrix Multiplication. *Linear Algebra and Its Applications*, **162–164**, 557–588, 1992.

[55] F. Le Gall, Faster Algorithms for Rectangular Matrix Multiplication. *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, 514–523, IEEE Computer Society Press, 2012.

[56] F. Le Gall, Powers of Tensors and Fast Matrix Multiplication. *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC 2014)*, 296–303, ACM Press, New York, 2014.

[57] G. Lotti, F. Romani, On the Asymptotic Complexity of Rectangular Matrix Multiplication. *Theoretical Computer Science*, **23**, 171–185, 1983.

[58] A. Massarenti, E. Raviolo, Corrigendum to "The rank of $n \times n$ matrix multiplication is at least $3n^2 - 2\sqrt{2}n^{3/2} - n$" [*Linear Algebra and its Applications*, **438, 11** (2013) 4500–4509]. *Linear Algebra and its Applications*, **445**, 369–371, 2014.

[59] T.S. Motzkin, Evaluation of Polynomials and Evaluation of Rational Functions. *Bull. of Amer. Math. Society*, **61**, 163, 1955.

[60] I. V. Oseledets, Approximation of Matrices with Logarithmic Number of Parameters. *Dokl. Math.*, **80, 2**, 653–654, 2009.

[61] I. V. Oseledets, Approximation of $2^d \times 2^d$ Matrices Using Tensor Decomposition. *SIAM J. on Matrix Analysis and Applications*, **31**, **4**, 2130–2145, 2010.

[62] I.V. Oseledets, E.E. Tyrtyshnikov, TT-cross Approximation for Multidimensional Arrays. *Linear Algebra Appls.* **432, 1**, 70–88, 2010.

[63] I.V. Oseledets, E.E. Tyrtyshnikov, Algebraic Wavelet Transform via Quantics Tensor Train Decomposition, *SIAM J. Scientific Computing*, **33, 3**, 1315–1328, 2011.

[64] A.M. Ostrowski, On Two Problems in Absract Algebra Connected with Horner's Rule. In the *Studies Presented to R. von Mises*, 40–48, Academic Press, New York, 1954.

[65] V.Y. Pan, On Methods of Computing the Values of Polynomials. *Uspekhi Matematicheskikh Nauk*, **21**, **1(127)**, 103–134, 1966. [Transl. *Russian Mathematical Surveys*, **21, 1(127)**, 105–137, 1966.]

[66] V.Y. Pan, On Schemes for the Evaluation of Products and Inverses of Matrices (in Russian). *Uspekhi Matematicheskikh Nauk*, **27**, **5 (167)**, 249–250, 1972.

[67] V.Y. Pan, Strassen's Algorithm Is Not Optimal. Trilinear Technique of Aggregating for Fast Matrix Multiplication. *Proc. the 19th Annual IEEE Symposium on Foundations of Computer Science (FOCS'78)*, 166–176, IEEE Computer Society Press, Long Beach, California, 1978.

[68] V.Y. Pan, Fields Extension and Trilinear Aggregating, Uniting and Canceling for the Acceleration of Matrix Multiplication. *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science (FOCS'79)*, 28–38, IEEE Computer Society Press, Long Beach, California, 1979.

[69] V.Y. Pan, New Fast Algorithms for Matrix Operations. *SlAM J. on Computing*, **9, 2**, 321–342, 1980, and Research Report RC 7555, *IBM T.J. Watson Research Center*, February 1979.

[70] V.Y. Pan, The Bit-Operation Complexity of the Convolution of Vectors and of the DFT. Technical report 80-6, Computer Science Dept., SUNYA, Albany, NY, 1980. (Abstract in Bulletin of EATCS, **14**, page 95, 1981.)

[71] V.Y. Pan, New Combinations of Methods for the Acceleration of Matrix Multiplications. *Computers and Mathematics (with Applications)*, **7, 1**, 73–125, 1981.

[72] V.Y. Pan, Trilinear Aggregating with Implicit Canceling for a New Acceleration of Matrix Multiplication. *Computers and Mathematics (with Applications)*, **8, 1**, 23–34, 1982.

[73] V.Y. Pan, How Can We Speed up Matrix Multiplication? *SIAM Review,* **26**, **3**, 393–415, 1984.

[74] V.Y. Pan, Trilinear Aggregating and the Recent Progress in the Asymptotic Acceleration of Matrix Operations. *Theoretical Computer Science*, **33**, 117–138, 1984.

[75] V.Y. Pan, *How to Multiply Matrices Faster. Lecture Notes in Computer Science*, **179**, Springer, Berlin, 1984.

[76] V.Y. Pan, Better Late Than Never: Filling a Void in the History of Fast Matrix Multiplication and Tensor Decompositions, arXiv:1411.1972, November 2014.

[77] R. L. Probert, On the Additive Complexity of Matrix Multiplication. *SIAM J. on Computing*, **5, 2**, 187–203, 1976.

[78] R.L. Probert, On the Complexity of Symmetric Computations. *Canadian J. 0f Information Processing and Operational Res.*, **12**, 71–86, 1974.

[79] F. Romani, private communication, 1979.

[80] F. Romani, Some Properties of Disjoint Sum of Tensors Related to MM. *SIAM J. on Computing*, **11, 2**, 283–267, 1982.

[81] P. Sankowski, M. Mucha, Fast Dynamic Transitive Closure with Lookahead. *Algorithmica*, **56, 2**, 180 197, 2010.

[82] A. Schönhage, Partial and Total Matrix Multiplication. *SIAM J. on Computing*, **10, 3**, 434–455, 1981.

[83] A. Schönhage, Asymptotically Fast Algorithms for the Numerical Multiplication and Division of Polynomials with Complex Coefficients. *Proc. EUROCAM, Marseille* (edited by J. Calmet), *Lecture Notes in Computer Science* **144**, 3–15, Springer, Berlin, 1982.

[84] A. Schönhage, V. Strassen, Schnelle Multiplikation grosse Zahlen. *Computing*, **7**, 281–292, 1971.

[85] A.V. Smirnov, The Bilinear Complexity and Practical Algorithms for Matrix Multiplication. *Computational Mathematics and Mathematical Physics*, **53, 12**, 1781–1795 (Pleiades Publishing, Ltd), 2013. Original Russian Text in *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, **53, 12**, 1970–1984, 2013.

[86] A. J. Stothers, On the Complexity of Matrix Multiplication. Ph.D. Thesis, University of Edinburgh, 2010.

[87] V. Strassen, Gaussian Elimination Is Not Optimal. *Numericshe Math.*, **13**, 354–356, 1969.

[88] V. Strassen, Vermeidung von Divisionen. *J. Reine Angew. Math.*, **264**, 184–202, 1973.

[89] V. Strassen, The Asymptotic Spectrum of Tensors and the Exponent of Matrix Multiplication. *Proc. 27th Ann. Symposium on Foundation of Computer Science*, 49–54, 1986.

[90] J. Todd, Motivation for Working in Numerical Analysis. *Communication on Pure and Applied Math.*, **8**, 97–116, 1955.

[91] E. E. Tyrtyshnikov, Tensor Approximations of Matrices Generated by Asymptotically Smooth Functions. *Mat. Sbornik*, **194, 6**, 147–160, 2003.

[92] V. Vassilevska Williams, Multiplying Matrices Faster than Coppersmith–Winograd. Version available at http://theory.stanford.edu/virgi/matrixmult-f.pdf, retrieved on January 30, 2014. Also see *Proc. 44th Annual ACM Symposium on Theory of Computing (STOC 2012)*, 887–898, ACM Press, New York, 2012.

[93] A. Waksman, On Winograd's Algorithm for Inner Products. *IEEE Transactions on Computers*, **C-19**, 360–361, 1970.

[94] S. Winograd, On the Number of Multiplications Required to Compute Certain Functions. *Proc. of the National Academy of Sciences*, **58**, 1840–1842, 1967.

[95] S. Winograd, A New Algorithm for Inner Product. *IEEE Transaction on Computers*, **C–17**, 693–694, 1968.

[96] S. Winograd, On the Number of Multiplications Necessary to Compute Certain Functions. *Communications on Pure and Applied Mathematics*, **23**, 165–179, 1970.

[97] S. Winograd, *Arithmetic Complexity of Computations*. CBMS-NSF Regional Conference Series in Applied Math., SIAM, Philadelphia, 1980.

[98] R. Yuster, U. Zwick, U. Detecting Short Directed Cycles Using Rectangular Matrix Multiplication and Dynamic Programming. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, 254–260, 2004.

[99] R. Yuster, U. Zwick, Fast Sparse Matrix Multiplication. *ACM Transactions on Algorithms*, **1, 1**, 2–13, 2005.

[100] U. Zwick. All-pairs Shortest Paths Using Bridging Sets and Rectangular Matrix Multiplication. *Journal of the ACM*, **49**, 289–317, 2002.