




# Matrix Multiplication Verification Using Coding Theory\*

Huck Bennett   

University of Colorado Boulder, CO, USA

Karthik Gajulapalli   

Georgetown University, Washington DC, USA

Alexander Golovnev   

Georgetown University, Washington DC, USA

Evelyn Warton  

Oregon State University, Corvallis, OR, USA

---

## Abstract

We study the *Matrix Multiplication Verification Problem* (MMV) where the goal is, given three  $n \times n$  matrices  $A$ ,  $B$ , and  $C$  as input, to decide whether  $AB = C$ . A classic randomized algorithm by Freivalds (MFCS, 1979) solves MMV in  $\tilde{O}(n^2)$  time, and a longstanding challenge is to (partially) derandomize it while still running in faster than matrix multiplication time (i.e., in  $o(n^\omega)$  time).

To that end, we give two algorithms for MMV in the case where  $AB - C$  is *sparse*. Specifically, when  $AB - C$  has at most  $O(n^\delta)$  non-zero entries for a constant  $0 \leq \delta < 2$ , we give (1) a deterministic  $O(n^{\omega-\varepsilon})$ -time algorithm for constant  $\varepsilon = \varepsilon(\delta) > 0$ , and (2) a randomized  $\tilde{O}(n^2)$ -time algorithm using  $\delta/2 \cdot \log_2 n + O(1)$  random bits. The former algorithm is faster than the deterministic algorithm of Künnemann (ESA, 2018) when  $\delta \geq 1.056$ , and the latter algorithm uses fewer random bits than the algorithm of Kimbrel and Sinha (IPL, 1993), which runs in the same time and uses  $\log_2 n + O(1)$  random bits (in turn fewer than Freivalds’s algorithm).

Our algorithms are simple and use techniques from coding theory. Let  $H$  be a parity-check matrix of a Maximum Distance Separable (MDS) code, and let  $G = (I \mid G')$  be a generator matrix of a (possibly different) MDS code in systematic form. Our deterministic algorithm uses fast rectangular matrix multiplication to check whether  $HAB = HC$  and  $H(AB)^T = H(C^T)$ , and our randomized algorithm samples a uniformly random row  $\mathbf{g}'$  from  $G'$  and checks whether  $\mathbf{g}'AB = \mathbf{g}'C$  and  $\mathbf{g}'(AB)^T = \mathbf{g}'C^T$ .

We additionally study the *complexity* of MMV. We first show that all algorithms in a natural class of deterministic linear algebraic algorithms for MMV (including ours) require  $\Omega(n^\omega)$  time. We also show a barrier to proving a super-quadratic running time lower bound for matrix multiplication (and hence MMV) under the Strong Exponential Time Hypothesis (SETH). Finally, we study relationships between natural variants and special cases of MMV (with respect to deterministic  $\tilde{O}(n^2)$ -time reductions).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms; Theory of computation  $\rightarrow$  Pseudorandomness and derandomization; Theory of computation  $\rightarrow$  Error-correcting codes

**Keywords and phrases** Matrix Multiplication Verification, Derandomization, Sparse Matrices, Error-Correcting Codes, Hardness Barriers, Reductions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.42

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2309.16176> [7]

---

\* Due to space constraints, we have made the body of our submission a modified version of the introduction to our paper. This introduction contains a detailed overview of our work and a comparison with prior work. Nevertheless, we strongly encourage the reader to read the full version of our paper [7].



© Huck Bennett, Karthik Gajulapalli, Alexander Golovnev, and Evelyn Warton; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 42; pp. 42:1–42:13



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Funding** *Huck Bennett*: Supported by NSF Grant CCF-2312297.

*Karthik Gajulapalli*: Supported by NSF grant CCF-2338730.

*Alexander Golovnev*: Supported by NSF grant CCF-2338730.

**Acknowledgements** We thank Amir Nayyeri for many helpful discussions in the early stages of work on this paper, and Mark Iwen [16] for answering questions about [17]. We also thank the anonymous reviewers for their helpful comments.

## 1 Introduction

The goal of the *Matrix Multiplication Problem* (MM) is to compute the product  $AB$  of two  $n \times n$  matrices  $A$  and  $B$  given as input. Matrix multiplication has many practical and theoretical applications, and because of this has been studied by an extensive line of work. The primary goal of this work has been to determine the running time  $O(n^\omega)$  of the fastest algorithms for MM, which is captured by the matrix multiplication exponent  $\omega$ .<sup>1</sup> The best upper bounds on  $\omega$  and related quantities continue to improve [23, 3, 11, 22, 25], and [25] recently showed the current best known bound of  $\omega \leq 2.371552$ . The dream of this line of work is to show that  $\omega = 2$ , and this in fact holds under certain plausible combinatorial and group-theoretic conjectures (see [10, Conjecture 4.7 and Conjecture 3.4]). Nevertheless, showing that  $\omega = 2$  seems very challenging for the time being.

In this work, we consider a variant of matrix multiplication where the goal is to *verify* that the product of two matrices is equal to a third matrix. Specifically, we study the *Matrix Multiplication Verification Problem* (MMV) where, given three  $n \times n$  matrices  $A$ ,  $B$ , and  $C$  as input, the goal is to decide whether  $AB = C$ . MMV is clearly no harder than matrix multiplication – it can be solved in  $O(n^\omega)$  time by computing the product  $AB$  and then comparing the product entry-wise against  $C$  – but it is natural to ask whether it is possible to do better. In what became classic work, Freivalds [12] answered this question in the affirmative and gave a simple, randomized algorithm that solves MMV in  $\tilde{O}(n^2)$  time. This  $\tilde{O}(n^2)$  running time bound is essentially the best possible, and so, unlike matrix multiplication, the complexity of MMV is relatively well understood.

However, it is in turn natural to ask whether it is possible to *derandomize* Freivalds’s algorithm partially or completely. More specifically, it is natural to ask whether it is possible to give a *deterministic* algorithm for MMV running in  $\tilde{O}(n^2)$  time or at least  $O(n^{\omega-\varepsilon})$  time for constant  $\varepsilon > 0$ .<sup>2</sup> Or, if it is not possible to give a deterministic algorithm for MMV with these running times, it is natural to ask whether it is possible to use fewer random bits than Freivalds’s algorithm, which uses  $n$  random bits. Trying to answer these questions has become a key goal for derandomization efforts, and has received substantial study [4, 19, 20, 21].

<sup>1</sup> Formally,  $\omega$  is defined as the infimum over  $\omega'$  such that the product of two  $n \times n$  matrices can be computed in  $O(n^{\omega'})$  time. So, MM algorithms are actually only guaranteed to run in  $O(n^{\omega+\varepsilon})$  time for any constant  $\varepsilon > 0$ .

<sup>2</sup> We use the notation  $\tilde{O}(f(n))$  to mean  $f(n) \cdot \text{poly}(\log f(n))$ . Freivalds’s algorithm uses  $O(n^2)$  arithmetic operations, each of which takes  $\text{poly}(\log n)$  time when working over integer matrices with entries bounded in magnitude by  $\text{poly}(n)$ ; we assume this setting in the introduction.

Of course, it is only possible for such a  $O(n^{\omega-\varepsilon})$ -time algorithm to exist if  $\omega > 2$ . We assume that this is the case throughout the introduction.

■ **Table 1** Algorithms for MMV on matrices  $A, B, C \in \mathbb{Z}^{n \times n}$  with entries of magnitude at most  $\text{poly}(n)$  and such that  $AB - C$  has at most  $n^\delta$  non-zero entries for  $0 \leq \delta \leq 2$ . Our new algorithms are shown in bold. We list asymptotic running times, with  $\text{poly}(\log n)$  factors suppressed for readability, and the number of random bits used to achieve success probability  $1/2$ . (Each of the three listed randomized algorithms has one-sided error, so this probability is meaningful.) Here  $\omega(\cdot, \cdot, \cdot)$  is the rectangular matrix multiplication exponent,  $\omega = \omega(1, 1, 1)$  is the (square) matrix multiplication exponent, and  $\varepsilon > 0$  is an arbitrarily small positive constant.

Algorithm	Asymptotic Runtime	Bits of Randomness
Matrix Multiplication	$n^{\omega+\varepsilon}$	0
Random Entry Sampling (folklore)	$n^{3-\delta}$	$2n^{2-\delta} \cdot \log_2(n) + O(1)$
Freivalds's Algorithm [12]	$n^2$	$n$
Vandermonde Mat. Sampling [19]	$n^2$	$\log_2(n) + O(1)$
Multipoint Poly. Evaluation [21]	$n^2 + n^{1+\delta}$	0
Cauchy Bound [20]	$n^3$ ( $n^2$ in Integer RAM)	0
<b>Parity Check/Fast RMM (Thm. 1)</b>	$n^{\omega(1,1,\delta/2)+\varepsilon}$	0
<b>Cauchy Mat. Sampling (Thm. 2)</b>	$n^2$	$\frac{\delta}{2} \cdot \log_2(n) + O(1)$

## 1.1 Our Results

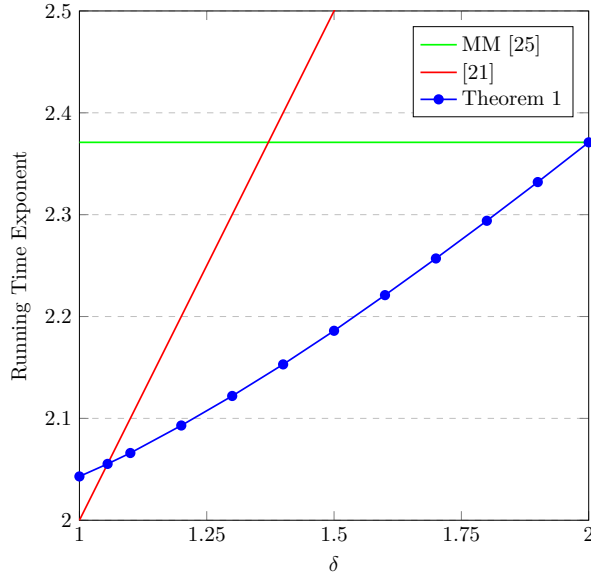
Our main results are two new algorithms for the Matrix Multiplication Verification Problem in the *sparse* regime, i.e., in the case where  $AB - C$  is promised to have few non-zero entries (if any). See Table 1 for a summary of our algorithms and how they compare to other known algorithms for MMV. Additionally, we give a barrier for giving a fast algorithm for MMV using a broad class of linear algebraic techniques, a barrier to showing hardness of MMV, and reductions between variants of MMV.

### 1.1.1 Algorithms

Besides being inherently interesting, MMV in the sparse regime is the natural decision version of the well-studied *Output-Sensitive Matrix Multiplication Problem* (OSMM). It is also motivated by the following scenario. Suppose that Alice wants to compute the product  $AB$  of two large matrices  $A$  and  $B$ , but has restricted computational resources. So, she sends  $A$  and  $B$  to Bob, who has more extensive computational resources. Bob computes the product  $AB$ , and sends the result back to Alice over a noisy channel (without error-correction, which increases the size of the message), from which Alice receives a matrix  $C$ . Alice knows that either  $C = AB$  as desired, or that  $C$  is corrupted but (with high probability) only differs from  $AB$  in a few entries. She wants to check which is the case efficiently.

We define  $\|\mathbf{v}\|_0$  (respectively,  $\|M\|_0$ ) to be the number of non-zero entries in (i.e., Hamming weight of) a vector  $\mathbf{v}$  (respectively, matrix  $M$ ). We call a vector  $\mathbf{v}$  (respectively, matrix  $M$ ) *t-sparse* if  $\|\mathbf{v}\|_0 \leq t$  (respectively, if  $\|M\|_0 \leq t$ ).

Our first algorithm (given in Figure 2) is deterministic, and uses fast rectangular matrix multiplication. For  $\alpha, \beta, \gamma \in [0, 1]$ , let the rectangular matrix multiplication exponent  $\omega(\alpha, \beta, \gamma)$  be the infimum over values  $\omega' > 0$  such that the product of a  $n^\alpha \times n^\beta$  matrix and a  $n^\beta \times n^\gamma$  matrix can be computed using  $O(n^{\omega'})$  arithmetic operations. Note that  $\omega = \omega(1, 1, 1)$  is the standard (square) matrix multiplication exponent.



■ **Figure 1** Running times of deterministic algorithms for MMV when  $AB - C$  is  $O(n^\delta)$ -sparse for  $1 \leq \delta \leq 2$ . Our algorithm from Theorem 1 is faster than the best known algorithms for matrix multiplication [25] and faster than Künnemann’s algorithm [21] for all  $1.056 \leq \delta < 2$ . The plotted blue points corresponding to the running time of the algorithm in Theorem 1 are derived from the bounds on  $\omega(1, 1, \delta/2)$  in [25, Table 1]. The line segments connecting them are justified by the fact that  $\omega(1, 1, \cdot)$  is a convex function.

- (1) Let  $k := \lceil \sqrt{t} \rceil$ .
- (2) Compute an arbitrary prime  $p$  that satisfies  $n \leq p \leq 2n$ .
- (3) Compute a (parity check) matrix  $H \in \mathbb{F}_p^{k \times n}$  of a code with distance at least  $k + 1$ .
- (4) Output YES if  $H(AB - C) = 0$  and  $H(AB - C)^T = 0$  (where arithmetic is performed over  $\mathbb{Z}$ ), and output NO otherwise.

■ **Figure 2** Deterministic Algorithm for MMV corresponding to Theorem 1.

► **Theorem 1** (Fast deterministic MMV for sparse matrices, informal). *Let  $A, B, C \in \mathbb{Z}^{n \times n}$  be matrices satisfying  $\max_{i,j} \{|A_{i,j}|, |B_{i,j}|, |C_{i,j}|\} \leq n^c$  for some constant  $c > 0$  and satisfying  $\|AB - C\|_0 \leq n^\delta$  for  $0 \leq \delta \leq 2$ . Then for any constant  $\varepsilon > 0$ , there is a deterministic algorithm for MMV on input  $A, B, C$  that runs in  $O(n^{\omega(1,1,\delta/2)+\varepsilon})$  time.*

We note that  $\omega(1, 1, \beta) < \omega$  for all  $\beta < 1$  (assuming  $\omega > 2$ );<sup>3</sup> and so our algorithm is faster than matrix multiplication when  $AB - C$  is promised to be  $O(n^\delta)$ -sparse for constant  $\delta < 2$ . Furthermore, it is faster than Künnemann’s algorithm [21], which is also for MMV in the regime where  $AB - C$  is sparse, when  $\omega(1, 1, \delta/2) < 1 + \delta$ . The equation  $\omega(1, 1, \delta/2) = 1 + \delta$  whose unique solution corresponds to the crossover point at which our algorithm becomes faster than Künnemann’s turns out to be relevant in other contexts too [27], and [23, 25] both provide bounds on its solution. Specifically, [25] shows that the solution  $\delta$  to this equation satisfies  $\delta \leq 1.056$ , and so our algorithm in Theorem 1 is (strictly) faster than any previously known deterministic algorithm for MMV when  $1.056 \leq \delta < 2$ .

<sup>3</sup> See [7, Theorem 2.3]

- (1) Let  $k := \lceil \sqrt{t}/\varepsilon \rceil$ .
- (2) Compute an arbitrary prime  $p$  satisfying  $k + n \leq p < 2(k + n)$ .
- (3) Compute a  $\lceil \sqrt{t} \rceil$ -regular matrix  $S = (\mathbf{s}_1, \dots, \mathbf{s}_k) \in \mathbb{F}_p^{n \times k}$ .
- (4) Sample a uniformly random column index  $i \sim \{1, \dots, k\}$ .
- (5) Output YES if  $AB\mathbf{s}_i = C\mathbf{s}_i$  and  $(AB)^T \mathbf{s}_i = C^T \mathbf{s}_i$  (where arithmetic is performed over  $\mathbb{Z}$ ), and output NO otherwise.

■ **Figure 3** Randomized Algorithm for MMV corresponding to Theorem 2.

Additional bounds on  $\omega(1, \delta/2, 1) = \omega(1, 1, \delta/2)$  – and hence the running time of the algorithm in Theorem 1 – appear in [25, Table 1]. For example, that table shows that  $\omega(1, 1, 0.55) < 2.067$  and  $\omega(1, 1, 0.95) < 2.333$  (which correspond to  $\delta = 1.1$  and  $\delta = 1.9$ , respectively). We also note that our algorithm runs in essentially optimal  $\tilde{O}(n^2)$  time when  $\delta \leq 0.642 \leq 2\omega^\perp$ , where  $\omega^\perp := \sup\{\omega' > 0 : \omega(1, 1, \omega') = 2\} \geq 0.321$  is the dual matrix multiplication exponent [25], but that Künnemann’s algorithm [21] runs in  $\tilde{O}(n^2)$  time for any  $\delta \leq 1$ .

Our second algorithm runs in  $\tilde{O}(n^2)$  time, but is randomized (see Figure 3). It uses few bits of randomness when  $AB - C$  is sparse.

► **Theorem 2** (Fast randomized MMV for sparse matrices, informal). *Let  $c > 0$  be a constant, let  $A, B, C \in \mathbb{Z}^{n \times n}$  be matrices satisfying  $\max_{i,j} \{|A_{i,j}|, |B_{i,j}|, |C_{i,j}|\} \leq n^c$  and satisfying  $\|AB - C\|_0 \leq n^\delta$  for  $0 \leq \delta \leq 2$ , and let  $\varepsilon = \varepsilon(n) \geq 1/n$ . Then there is a randomized algorithm for MMV on input  $A, B, C$  that runs in  $\tilde{O}(n^2)$  time, succeeds with probability  $1 - \varepsilon$ , and uses at most  $\lceil \delta/2 \cdot \log_2(n) + \log_2(1/\varepsilon) \rceil$  bits of randomness.*

Theorem 2 improves on the number of random bits used by the algorithm of Kimbrel and Sinha [19] when  $\delta < 2$  (which uses  $\log_2(n) + \log_2(1/\varepsilon) + O(1)$  random bits regardless of the sparsity of  $AB - C$ ), and matches the number of random bits used by their algorithm when  $\delta = 2$ . The algorithms both run in  $\tilde{O}(n^2)$  time. In fact, one may think of the algorithm summarized in Theorem 2 as a natural extension of the algorithm in [19] to handle the sparse case more efficiently, although it requires additional techniques to implement. (Our algorithm requires matrices with a stronger pseudorandomness property than theirs; see the “algorithmic techniques” section below.)

We note that Theorem 2 only improves on known algorithms when  $1 < \delta < 2$ , and only by a factor of  $\delta/2$ . Indeed, as mentioned above, when  $\delta \leq 1$  Künnemann’s algorithm [21] solves MMV *deterministically* in  $\tilde{O}(n^2)$  time, and when  $\delta = 2$  our algorithm matches the number of random bits used by Kimbrel and Sinha’s algorithm. Although seemingly modest, this constant-factor improvement is not surprising: any super-constant improvement on the number of bits used by [19] (i.e., an MMV algorithm using  $o(\log n)$  random bits) could be turned into a deterministic algorithm for MMV with only a sub-polynomial (i.e.,  $n^{o(1)}$ ) multiplicative increase in running time.

### 1.1.1.1 Algorithmic techniques

Here we briefly summarize the techniques that we use for the MMV algorithms corresponding to Theorems 1 and 2. We start by remarking that Theorems 1 and 2 hold not just for matrices over  $\mathbb{Z}$  with entries of polynomial magnitude, but also for matrices over all finite

fields  $\mathbb{F}_q$  with  $q \leq \text{poly}(n)$ .<sup>4</sup> In fact, our algorithms work “natively” in the finite field setting – i.e., on  $n \times n$  matrices  $A, B, C$  over finite fields  $\mathbb{F}_q$  – which is directly amenable to using techniques from coding theory. We assume this setting in the description below. Furthermore, there is a linear-time, sparsity-preserving reduction from MMV to the special case of MMV where  $C$  is fixed as  $C = 0$  and the goal is to decide whether  $AB = 0$  for input matrices  $A, B$  (see [21, Proposition 3.1]). We will also generally assume this setting in the introduction.

For both algorithms, we will use the observation that if  $AB - C$  is non-zero and  $t$ -sparse then at least one row or column of  $AB - C$  must be non-zero and  $k$ -sparse for  $k := \lfloor \sqrt{t} \rfloor$ . A similar observation appears in [26].

Our first, deterministic algorithm (Theorem 1) uses a matrix  $H$  over  $\mathbb{F}_q$  such that any  $k$  columns of  $H$  are linearly independent. Equivalently, we require a matrix  $H \in \mathbb{F}_q^{m \times n}$  such that for all non-zero vectors  $\mathbf{x} \in \mathbb{F}_q^n$  with  $\|\mathbf{x}\|_0 \leq k$  (corresponding to a sparse, non-zero column or row of  $AB$ ),  $H\mathbf{x} \neq \mathbf{0}$ . This is exactly the property guaranteed by the parity-check matrix  $H$  of an error correcting code  $\mathcal{C} \subseteq \mathbb{F}_q^n$  with minimum distance  $d > k$ . Moreover, if a code  $\mathcal{C}$  with minimum distance  $d = k + 1$  is a so-called Maximum Distance Separable (MDS) code, then it has a  $k \times n$  parity-check matrix  $H$ . MDS codes with useful parameters exist and have efficiently constructible parity-check matrices. In particular, (generalized) Reed-Solomon codes are MDS codes, and exist when  $k \leq n \leq q$  (see, e.g., [14]). Their parity-check matrices  $H$  are Vandermonde matrices, which are constructible in  $kn \cdot \text{poly}(\log q) \leq n^2 \cdot \text{poly}(\log q)$  time.

Our algorithm then uses fast rectangular matrix multiplication to compute  $HAB = (HA)B$  and  $H(AB)^T = (HB^T)A^T$  using roughly  $n^{\omega(1,1,\delta/2)}$  arithmetic operations, where  $0 \leq \delta \leq 2$  is such that  $t \leq n^\delta$ . If  $AB = 0$ , then  $HAB = H(AB)^T = 0$ . On the other hand, if  $AB \neq 0$  then  $AB$  is  $t$ -sparse and therefore has a  $k$ -sparse row or column. So, at least one of the expressions  $HAB$  and  $H(AB)^T$  is non-zero.

Our second, randomized algorithm (Theorem 2) uses a matrix  $S \in \mathbb{F}_q^{m \times n}$  with the property that all of its  $k \times k$  submatrices are non-singular. Matrices  $S$  with this property are called  $k$ -regular, and matrices  $S$  all of whose square submatrices (of any size) are non-singular are called *super regular*.<sup>5</sup> We note that  $k$ -regularity is stronger than the property we require for  $H$  in the first algorithm. In particular, if a matrix  $S \in \mathbb{F}_q^{m \times n}$  is  $k$ -regular and  $0 < \|\mathbf{x}\|_0 \leq k$ , then  $\|S\mathbf{x}\|_0 \geq m - k + 1$ . I.e.,  $S$  being  $k$ -regular implies not only that  $S\mathbf{x}$  is non-zero, but that  $S\mathbf{x}$  has relatively high Hamming weight for such  $\mathbf{x}$ . This property is useful because it implies that  $\Pr[\langle \mathbf{s}, \mathbf{x} \rangle \neq 0] \geq (m - k + 1)/m$ , where  $\mathbf{s}$  is a random row of  $S$ . Indeed, this observation leads to our second algorithm: we sample a random row  $\mathbf{s}$  from a  $k$ -regular matrix  $S \in \mathbb{F}_q^{m \times n}$  and check whether  $\mathbf{s}AB = \mathbf{0}$  and  $\mathbf{s}(AB)^T = \mathbf{0}$ . Setting, e.g.,  $m = 2k$ , we get that this algorithm succeeds with probability at least  $(2k - k + 1)/(2k) > 1/2$ .

It remains to construct (rows of)  $k$ -regular matrices  $S$  efficiently. Although a priori it is not even obvious that  $k$ -regular matrices exist for arbitrary  $k$ , in fact *super regular* matrices exist and are efficiently constructible. Specifically, we use a family of super regular (and hence  $k$ -regular) matrices called *Cauchy matrices*; the entries of such a matrix  $S$  are defined as  $S_{i,j} = 1/(x_i - y_j)$ , where  $x_1, \dots, x_m, y_1, \dots, y_n$  are distinct elements of  $\mathbb{F}_q$ . In fact, as follows from their definition, given a (random) index  $1 \leq i \leq m$ , it is even possible to construct the  $i$ th row of a Cauchy matrix  $S$  efficiently without computing the rest of the matrix, as needed.

<sup>4</sup> The algorithms also work over larger finite fields, but with slower running times due to the increased bit complexity of performing arithmetic operations over those fields.

<sup>5</sup> For formal definitions see [7, Section 2.4]

Finally, we remark that there is a deep connection between MDS codes and super regular matrices (and between generalized Reed-Solomon codes, Vandermonde matrices, and Cauchy matrices). Specifically, if  $G = (I \mid S)$  is the generator matrix of an MDS code in systematic form, then  $S$  is a super regular matrix [24]. Moreover, if such a matrix  $G$  is the generator matrix of a generalized Reed-Solomon code, then  $S$  is a Cauchy matrix [24].

### 1.1.2 Barriers

The dream for the line of work described in this paper is to give a deterministic,  $\tilde{O}(n^2)$ -time algorithm for MMV on arbitrary matrices. However, achieving this goal has proven to be very difficult despite a substantial amount of work towards it. So, it is natural to ask whether perhaps no such algorithm exists, i.e., whether MMV is in some sense *hard*. We first show a result in this direction, and then show a barrier result to showing SETH hardness of MMV (and even MM).<sup>6</sup>

#### 1.1.2.1 Linear algebraic algorithms barrier

We first prove that a natural class of deterministic linear algebraic algorithms for MMV based on multiplying smaller matrices – including the algorithm in Theorem 1 – cannot run in less than  $n^\omega$  time when using a matrix multiplication subroutine running in worst-case rectangular matrix multiplication time and when performing all multiplications independently. Specifically, the  $\Omega(n^\omega)$  lower bound holds if for all  $\alpha, \beta \geq 0$ , the subroutine requires  $\Omega(n^{\omega(1,1,\alpha)})$  to compute the product of an  $n \times n^\alpha$  matrix and an  $n \times n$ , and  $\Omega(n^{\omega(1,1,\beta)})$  time to compute the product of an  $n \times n$  matrix and an  $n \times n^\beta$  matrix.

The idea is that natural algorithms for verifying that  $AB = C$  for  $n \times n$  matrices  $A, B, C$  including ours amount to performing  $k$  “zero tests.” More specifically, the  $i$ th such test checks that  $L_i(AB - C)R_i = 0$  for some fixed  $n^{\alpha_i} \times n$  matrix  $L_i$  and  $n \times n^{\beta_i}$  matrix  $R_i$ , where  $\alpha_i, \beta_i \in [0, 1]$ . We observe that the conditions  $L_i(AB - C)R_i = 0$  for  $i = 1, \dots, k$  together correspond to a homogeneous system of  $\sum_{i=1}^k n^{\alpha_i + \beta_i}$  linear equations in the  $n^2$  variables corresponding to the entries of  $X = AB - C$  for  $1 \leq i, j \leq n$ . So, for this system to have  $X_{i,j} = 0$  for  $1 \leq i, j \leq n$  as its unique solution, it must be the case that  $\sum_{i=1}^k n^{\alpha_i + \beta_i} \geq n^2$ , which we show implies that  $\sum_{i=1}^k n^{\omega(1,1,\min(\alpha_i,\beta_i))} \geq \sum_{i=1}^k n^{\omega(\alpha_i,1,\beta_i)} \geq n^\omega$ . Therefore, an algorithm that independently computes each product  $L_i A B R_i$  in time  $\Omega(n^{\omega(1,1,\min(\alpha_i,\beta_i))})$  uses  $\Omega(n^\omega)$ .<sup>7</sup>

#### 1.1.2.2 A barrier to SETH-hardness of MM

While under certain reasonable conjectures, the matrix multiplication exponent  $\omega = 2$  (see [10, Conjecture 4.7 and Conjecture 3.4]), the best provable upper bound we have is  $\omega < 2.371552$  by [25]. Nevertheless, given the apparent difficulty of showing  $\omega \approx 2$ , it is natural to ask whether MM is in fact *hard*. To that end, we study showing its hardness under the *Strong Exponential Time Hypothesis* (SETH). However, rather than showing SETH-hardness of MM, we show a *barrier* to proving  $n^\gamma$ -hardness of MM for constant  $\gamma > 2$  under SETH. (Because MMV is trivially reducible to MM, our hardness barrier result also applies to MMV.)

<sup>6</sup> More properly, our first result is a barrier to giving a fast algorithm for MMV, and our second result is a barrier to showing hardness of MMV (i.e., it “gives a barrier to giving a barrier” for a fast MMV algorithm).

<sup>7</sup> For a more detailed exposition of this barrier see [7, Section 3.3]



We informally define several concepts used in the statement of our result. SETH says that for large constant  $k$ ,  $k$ -SAT instances on  $n$  variables take nearly  $2^n$  time to solve, and the *Nondeterministic Strong Exponential Time Hypothesis* (NSETH) says that certifying that such  $k$ -SAT formulas are *not* satisfiable takes nearly  $2^n$  time even for nondeterministic algorithms. We call a matrix *rigid* if the Hamming distance between it and all low-rank matrices is high (the Hamming distance and rank are quantified by two parameters). Rigid matrices have many connections to complexity theory and other areas, and a key goal is to find explicit, deterministic constructions of such matrices.

Intuitively, NSETH rules out showing hardness of problems with non-trivial co-nondeterministic algorithms under SETH. Somewhat more precisely, assuming NSETH, problems contained in  $\text{coTIME}[f(n)]$  (but perhaps only known to be in  $\text{TIME}[g(n)]$  for  $g(n) = \omega(f(n))$ ), cannot be shown to be  $\Omega(f(n)^{1+\varepsilon})$ -hard under SETH.<sup>8</sup> Künnemann [21] noted that, because Freivald’s algorithm shows that MMV is in  $\text{coTIME}[n^2 \cdot \text{poly}(\log n)]$ , NSETH rules out showing  $\Omega(n^\gamma)$  hardness of MMV under SETH for constant  $\gamma > 2$ .

In this work, we extend this observation and give a barrier not only to showing SETH-hardness of MMV but to showing hardness of MM. We demonstrate that, if there exists a constant  $\gamma > 2$  and a reduction from  $k$ -SAT to MM such that a  $O(n^{\gamma-\varepsilon})$ -time algorithm for MM for any constant  $\varepsilon > 0$  breaks SETH, then either (1) the *Nondeterministic Strong Exponential Time Hypothesis* (NSETH) is false, or (2) a new non-randomized algorithm for computing (arbitrarily large) rigid matrices exists. We also note that, by known results, falsifying NSETH implies a new circuit lower bound as a consequence. In short, our barrier result says that showing  $n^\gamma$ -hardness of MM under SETH for  $\gamma > 2$  would lead to major progress on important questions in complexity theory.<sup>9</sup>

A key idea that we use for proving our result is that it is possible to compute the product of two *non-rigid* matrices efficiently using a *nondeterministic* algorithm. This follows from two facts. First, by definition, a non-rigid matrix is the sum of a low-rank matrix  $L$  and a sparse matrix  $S$ , and using nondeterminism it is possible to guess  $L$  and  $S$  efficiently. Second, it is possible to compute the product of two sparse matrices or a low-rank matrix and another matrix efficiently. (In fact, we also use nondeterminism to guess a *rank factorization* of  $L$ , and this factorization is what allows for fast multiplication by  $L$ .)

Very roughly, we prove the barrier result as follows. We first suppose that there is a reduction from  $k$ -SAT to (potentially multiple instances of) matrix multiplication. In particular, such a reduction outputs several pairs of matrices to be multiplied. We then analyze three cases:

1. If the matrices output by this reduction always have small dimension (as a function of  $n$ ), then we can compute the product of each pair quickly using standard matrix multiplication algorithms (even using naïve, cubic-time matrix multiplication). This leads to a fast, deterministic algorithm for  $k$ -SAT, which refutes SETH (and hence NSETH).
2. If the matrices output by this reduction are always *not* rigid, then we can compute the product of each pair quickly using the nondeterministic algorithm sketched above. This leads to a fast, nondeterministic algorithm for showing that  $k$ -SAT formulas are not satisfiable, which refutes NSETH.
3. Finally, if neither of the above cases holds, then the reduction must sometimes output rigid matrices with large dimension as a function of  $n$ . So, we obtain an algorithm for generating arbitrarily large rigid matrices using an NP oracle: iterate through all  $k$ -SAT

<sup>8</sup> We refer the reader to see [7, Definition 2.22] for a formal definition of SETH-hardness.

<sup>9</sup> See see [7, Section 4] for a more detailed exposition.



formulas  $\varphi$  with at most a given number of variables, apply the reduction from  $k$ -SAT to MM to each formula, and then use the NP oracle to check whether each large matrix output by the reduction is rigid.

We remark that although NSETH is a strong and not necessarily widely believed conjecture, [18, 9] showed that refuting it (as in Item 2 above) would nevertheless imply an interesting new circuit lower bound. Specifically, they showed that if NSETH is false, then the complexity class  $E^{NP}$  requires series-parallel circuits of size  $\omega(n)$ .

Additionally, we remark that despite how slow the “iterate through all sufficiently large  $k$ -SAT formulas and apply the  $k$ -SAT-to-MM reduction to each one” algorithm described in Item 3 seems, it would still substantially improve on state-of-the-art non-randomized algorithms for generating rigid matrices. This is also true despite the fact that the algorithm uses an NP oracle.<sup>10</sup>

### 1.1.3 Reductions

Again, motivated by the apparent challenge of fully derandomizing Freivalds’s algorithm, we study relationships between variants of MMV with the goal of understanding what makes the problem hard to solve deterministically in  $\tilde{O}(n^2)$  time but easy to solve in  $\tilde{O}(n^2)$  time using randomness (in contrast to MM). More specifically, we study which variants are potentially easier than MMV (i.e., reducible to MMV, but not obviously solvable deterministically in  $\tilde{O}(n^2)$  time using known techniques), equivalent to MMV, and potentially harder than MMV (i.e., variants to which MMV is reducible, but which are not obviously as hard as MM). We study these questions by looking at deterministic  $\tilde{O}(n^2)$ -time reductions between variants. See Figure 4 for a summary of our results.

First, we show that two apparently special cases of MMV are in fact equivalent to MMV. These special cases are: (1) the *Inverse Verification Problem*, where the goal is to verify that  $B = A^{-1}$  for input matrices  $A$  and  $B$  (equivalently, the special case of MMV where  $C = I_n$ ), and (2) the *Symmetric MMV Problem*, where the input matrices  $A$  and  $B$  (but not necessarily  $C$ ) are symmetric.<sup>11</sup> These reductions are relatively simple, and complement the (also simple) reduction of [21], who showed that the All Zeroes Problem (i.e., the special case of MMV where  $C = 0$ ) is MMV-complete.

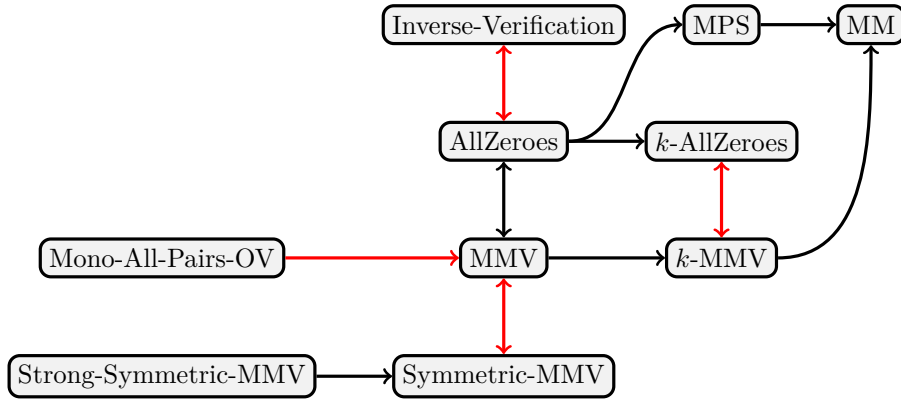
Second, we identify two problems that are  $\tilde{O}(n^2)$ -time reducible to MMV, but are not clearly solvable in  $\tilde{O}(n^2)$  time or equivalent to MMV. These problems are: (1) the *Strong Symmetric MMV Problem*, where all three of the input matrices  $A$ ,  $B$ , and  $C$  are symmetric, and (2) the Monochromatic All Pairs Orthogonal Vectors Problem, where the goal is, given vectors  $\mathbf{a}_1, \dots, \mathbf{a}_n$  to decide whether  $\langle \mathbf{a}_i, \mathbf{a}_j \rangle = 0$  for all  $i \neq j$ .

Third, we identify two problems for which there are  $\tilde{O}(n^2)$ -time reductions from MMV and that are  $\tilde{O}(n^2)$ -time reducible to MM. These “MMV/MM-intermediate problems” are: (1) the Matrix Product Sparsity Problem (MPS), in which the goal is, given matrices  $A$  and  $B$  and  $r \geq 0$  as input, to decide whether  $\|AB\|_0 \leq r$ , and (2) the  $k$ -MMV problem, in which given matrices  $A_1, \dots, A_k, C$  as input, the goal is to decide whether  $\prod_{i=1}^k A_i = C$ . We note that MPS is equivalent to the counting version of the Orthogonal Vectors Problem ( $\#OV$ ).<sup>12</sup> We additionally show that  $k$ -MMV is equivalent to the  $k$ -All Zeroes problem,

<sup>10</sup> See [7, Section 4] for a more thorough discussion.

<sup>11</sup> See [7, Section 5] for the complete reductions

<sup>12</sup> Indeed, Monochromatic All Pairs Orthogonal Vectors is no harder than MMV (and not obviously equivalent), Bichromatic All Pairs Orthogonal Vectors is equivalent to the All Zeroes Problem and is



■ **Figure 4** A diagram of reductions among MMV and related problems on  $n \times n$  matrices. Arrows represent deterministic  $O(n^2)$ -time reductions (and double-headed arrows denote equivalences under such reductions). Red arrows indicate new (non-trivial) reductions.

i.e.,  $k$ -MMV where  $C$  is fixed to be 0. See Figure 4 for a summary of these variants and reductions between them. For a full presentation of definitions and reductions we refer the reader to [7, Section 5].

## 1.2 Related Work

We next discuss other algorithms for MMV and related problems on  $n \times n$  integer matrices  $A$ ,  $B$ , and  $C$ . We summarize these algorithms, as well as ours, in Table 1. We start by noting that it suffices to consider the special case of MMV where  $C = 0$  (i.e., where the goal is to decide whether  $AB = 0$ ), which is called the All Zeroes Problem. Indeed, a result from [21] shows that there is a simple  $O(n^2)$ -time reduction from MMV on  $n \times n$  matrices  $A, B, C$  to the All Zeroes problem on  $2n \times 2n$  matrices  $A', B'$  with the property that  $\|AB - C\|_0 = \|A'B'\|_0$ . So, for this section we consider the All Zeroes Problem without loss of generality.

Perhaps the most closely related works to ours are [17, 1], which use fast rectangular matrix multiplication for the *Output-Sensitive Matrix Multiplication Problem* (OSMM). In  $t$ -OSMM, the goal is, given matrices  $A, B$  as input, to compute the product  $AB$  when it is promised to be  $t$ -sparse. There is a trivial reduction from MMV when the output is promised to be  $t$ -sparse to  $t$ -OSMM – compute  $AB$  and check whether it is equal to 0. Indeed, OSMM is essentially the search version of sparse MMV. In particular, [17] use techniques from compressive sensing to solve OSMM. However, it is not clear that the measurement matrix  $M$  in [17] can be constructed deterministically in  $\tilde{O}(n^2)$  time, and so the algorithm in [17] is a non-uniform algorithm as described. There are other candidate measurement matrices with deterministic constructions that may work for a similar purpose [16], but the exact tradeoffs do not seem to have been analyzed and it is not clear that it is possible to get a (uniform) algorithm with the same parameters. Additionally, [17] only handles the case when all columns or rows of  $AB$  are promised to have a given sparsity, rather than the case where there is a “global bound” of  $t$  on the sparsity of the matrix product itself.

---

therefore equivalent to MMV, and MPS/#OV is at least as hard as MMV. In the fine-grained complexity setting OV variants are usually considered with  $n$  vectors in dimension  $d = \text{poly}(\log n)$ ; here we are considering the regime where  $d = n$ .

The main algorithm in [1] for OSMM (summarized in [1, Theorem 1.4]) runs in randomized time  $O(n^{1.3459\delta})$  when both the input matrices  $A, B$  and their product  $AB$  are  $n^\delta$ -sparse.<sup>13</sup> For the special case when all entries in  $A, B$  are non-negative [1] give a deterministic algorithm with the same running time as their randomized algorithm. We note that [1] was written independently and concurrently with this work.

Besides simply using matrix multiplication, perhaps the most natural idea for an algorithm for the All Zeros problem is to compute a random entry  $(AB)_{i,j}$  of  $AB$  and check whether it is non-zero. If  $\|AB\|_0 \geq n^\delta$ , then sampling, say,  $10n^{2-\delta}$  random entries of  $AB$  independently will find a non-zero entry with good constant probability. Because computing each such entry amounts to computing an inner product, and sampling indices  $i, j \sim \{1, \dots, n\}$  takes roughly  $2 \log_2 n$  random bits, this algorithm overall takes  $\tilde{O}(n^{3-\delta})$  time and  $O(n^{2-\delta} \log n)$  random bits. So, this algorithm is relatively efficient and succeeds with good probability in the case when  $AB$  is dense, but even then requires a relatively large number of random bits. We also note the somewhat odd fact that this algorithm is most efficient when  $AB$  is dense, whereas our algorithms are most efficient when  $AB$  is sparse.

Freivalds's algorithm [12] works by sampling a uniformly random vector  $\mathbf{x} \sim \{0, 1\}^n$ , and outputting "YES" if  $AB\mathbf{x} = \mathbf{0}$  and "NO" otherwise. If  $AB = \mathbf{0}$ , then this algorithm is always correct, and if  $AB \neq \mathbf{0}$  then it fails with probability at most  $1/2$ .<sup>14</sup> In particular, Freivalds's algorithm has one-sided error with no false negatives (i.e., it is a **coRP** algorithm).

A key idea for subsequent algorithms was to reduce MMV to a question about polynomials. The main idea is the following. Define  $\mathbf{x} := (1, x, x^2, \dots, x^{n-1})^T$ , where  $x$  is an indeterminate, and define  $p_i(x) := (AB\mathbf{x})_i = \sum_{j=1}^n (AB)_{i,j} \cdot x^{j-1}$ . Note that  $AB = \mathbf{0}$  if and only if the polynomials  $p_i(x)$  are identically zero (as formal polynomials) for all  $i \in \{1, \dots, n\}$ . Furthermore, if the  $i$ th row of  $AB$  is non-zero then  $p_i(x)$  is a non-zero polynomial of degree at most  $n-1$ , and therefore has at most  $n-1$  distinct complex (and hence integral) roots. So, for such  $p_i(x)$  and a non-empty set  $S \subset \mathbb{Z}$ ,  $\Pr_{\alpha \sim S}[p(\alpha) = 0] \leq (n-1)/|S|$ , which is less than  $1/2$  when  $|S| \geq 2n$ . This observation leads to the following algorithm for MMV, which forms the basis for Kimbrel and Sinha's algorithm [19]. Sample  $\alpha \sim \{1, \dots, 2n\}$ , and output "YES" if and only if  $AB\boldsymbol{\alpha} = \mathbf{0}$  for  $\boldsymbol{\alpha} := (1, \alpha, \alpha^2, \dots, \alpha^{n-1})^T$ . Using associativity, it is possible to compute this product as  $A(B\boldsymbol{\alpha})$  using  $O(n^2)$  arithmetic operations.

However, there is an issue with this algorithm: it requires computing powers of  $\alpha$  up to  $\alpha^{n-1}$ . These powers require  $\Omega(n)$  bits to represent for any integer  $\alpha \geq 2$ , and so performing arithmetic operations with them takes  $\Omega(n)$  time. To solve this, Kimbrel and Sinha instead consider the "test vector"  $\boldsymbol{\alpha}$  modulo an (arbitrary) prime  $2n \leq q \leq 4n$ , which they can find deterministically in  $O(n^2)$  time. They show that their algorithm is still correct with good probability (over the choice of  $\alpha$ ) with this modification.

Korec and Wiedermann [20] showed how to *deterministically* find a good  $\alpha$  for the above test – that is, a value  $\alpha$  such that  $p_i(\alpha) \neq 0$  if  $p_i$  is not identically zero – using *Cauchy's bound*, which gives an upper bound on the magnitude of the largest root of a polynomial as a function of the polynomial's coefficients. Namely, they just choose  $\alpha$  larger than Cauchy's bound. (They note that the maximum magnitude of an entry in  $AB$  – and hence of a coefficient in

<sup>13</sup>A more general version of this theorem, which gives an algorithm whose running time depends both on the sparsity of the input matrices  $A, B$  and of their product  $AB$ , appears as [1, Theorem 1.7].

<sup>14</sup>To see this, note that in the latter case some row  $\mathbf{s}^T$  of  $AB$  must be non-zero, and let  $j^*$  be the index of the last non-zero entry in  $\mathbf{s}$ . Then for uniformly random  $\mathbf{x} \sim \{0, 1\}^n$ ,  $\Pr[AB\mathbf{x} = \mathbf{0}] \leq \Pr[(\mathbf{s}, \mathbf{x}) = 0] = \Pr[s_{j^*}x_{j^*} = -\sum_{k=1}^{j^*-1} s_k x_k] \leq 1/2$ . Moreover, this holds for matrices  $A, B$  over any ring  $R$ , and so Freivalds's algorithm works for MMV over any ring  $R$ .

any of the polynomials  $p_i(x)$  – is at most  $n\mu^2$ , where  $\mu$  is the maximum magnitude of an entry in  $A$  or  $B$ .) Their algorithm uses only  $O(n^2)$  arithmetic operations, but again requires computing powers of  $\alpha$  up to  $\alpha^{n-1}$ , and therefore the algorithm has bit complexity  $\Omega(n^3)$ .

Additionally, we mention the work of Künnemann [21], which works for MMV over finite fields  $\mathbb{F}_q$  with  $q > n^2$  (he reduces MMV over the integers to MMV over such fields). His algorithm works by considering the bivariate polynomial  $f(x, y) = f_{A,B}(x, y) := \mathbf{x}^T A B \mathbf{y}$  for  $\mathbf{x} = (1, x, x^2, \dots, x^{n-1})$ ,  $\mathbf{y} = (1, y, y^2, \dots, y^{n-1})$ , where  $x$  and  $y$  are indeterminates, and the corresponding univariate polynomial  $g(x) = g_{A,B}(x) := f(x, x^n)$ . The coefficient of  $x^{(i-1)+(j-1)n}$  in  $g(x)$  (and of  $x^{i-1}y^{j-1}$  in  $f(x, y)$ ) is equal to  $(AB)_{i,j}$ , and so to decide whether  $AB = 0$  it suffices to decide whether  $g(x)$  (or  $f(x, y)$ ) is identically zero as a formal polynomial.<sup>15</sup> He shows that to do this it in turn suffices to decide whether  $g(\alpha^i) = 0$  for all  $i \in \{0, \dots, t-1\}$ , where  $\alpha \in \mathbb{F}_q$  is an element of order at least  $n^2$  and  $t = n^\delta$  is an upper bound on the sparsity of  $AB$ . Indeed, he notes that the system of equations  $g(1) = \dots = g(\alpha^{t-1}) = 0$  is a Vandermonde system of homogeneous linear equations in the at most  $t$  non-zero entries  $(AB)_{i,j}$  in  $AB$ , and so its only solution is the solution  $(AB)_{i,j} = 0$  for all  $1 \leq i, j \leq n$  (i.e., it must be the case that  $AB = 0$ ). To evaluate  $g$  on the  $t$  values  $1, \alpha, \dots, \alpha^{t-1}$  quickly, he uses a known result about fast multipoint polynomial evaluation.

We also note that MMV and its variants have also been studied from angles other than derandomization of Freivalds’s algorithm. Notably, [8] gave a  $O(n^{5/3})$ -time *quantum* algorithm for MMV, [15] studied the *Boolean* Matrix Multiplication Verification problem, and [13, 26] study the problem of *correcting* matrix products. I.e., they study the problem of *computing*  $AB$  given matrices  $A$ ,  $B$ , and  $C$  where  $\|AB - C\|_0$  is guaranteed to be small, which Künnemann showed is equivalent to OSMM.

Finally, we remark that other recent works including [9, 5, 2, 6] have also studied “barriers to SETH hardness”.

### 1.3 Open Questions

Of course, the main question that we leave open is whether Freivalds’s algorithm can be fully derandomized, i.e., whether there is a deterministic  $\tilde{O}(n^2)$ -time algorithm for MMV on  $n \times n$  matrices over finite fields  $\mathbb{F}_q$  with  $q \leq \text{poly}(n)$  and integer matrices with entries  $[-n^c, n^c]$  for constant  $c > 0$ . Additionally, it would be interesting to extend our results for MMV in the sparse regime to Output Sensitive Matrix Multiplication. The coding-theoretic techniques that we use seem amenable to this.

---

#### References

- 1 Amir Abboud, Karl Bringmann, Nick Fischer, and Marvin Künnemann. The time complexity of fully sparse matrix multiplication. In *SODA*, 2024.
- 2 Divesh Aggarwal, Huck Bennett, Zvika Brakerski, Alexander Golovnev, Rajendra Kumar, Zeyong Li, Spencer Peters, Noah Stephens-Davidowitz, and Vinod Vaikuntanathan. Lattice problems beyond polynomial time. In *STOC*, 2023.
- 3 Joah Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA*, 2021.
- 4 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost  $k$ -wise independent random variables. In *FOCS*, 1990.

---

<sup>15</sup> Indeed, [21] notes that this mapping from  $A, B$  to  $g(x)$  is a reduction from the All Zeroes Problem to Univariate Polynomial Identity Testing (UPIT).

- 5 Tatiana Belova, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, and Denil Sharipov. Polynomial formulations as a barrier for reduction-based hardness proofs. In *SODA*, 2023.
- 6 Tatiana Belova, Alexander S. Kulikov, Ivan Mihajlin, Olga Ratseeva, Grigory Reznikov, and Denil Sharipov. Computations with polynomial evaluation oracle: ruling out superlinear SETH-based lower bounds. *arXiv*, 2023. [arXiv:2307.11444](https://arxiv.org/abs/2307.11444).
- 7 Huck Bennett, Karthik Gajulapalli, Alexander Golovnev, and Evelyn S Warton. Matrix multiplication verification using coding theory. *arXiv preprint*, 2023. [arXiv:2309.16176](https://arxiv.org/abs/2309.16176).
- 8 Harry Buhrman and Robert Spalek. Quantum verification of matrix products. *arXiv*, 2004. [arXiv:quant-ph/0409035](https://arxiv.org/abs/quant-ph/0409035).
- 9 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *ITCS*, 2016.
- 10 Henry Cohn, Robert Kleinberg, Balázs Szegedy, and Christopher Umans. Group-theoretic algorithms for matrix multiplication. In *FOCS*, 2005.
- 11 Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *FOCS*, 2023.
- 12 Rūsiņš Freivalds. Fast probabilistic algorithms. In *MFCS*, 1979.
- 13 Leszek Gasieniec, Christos Levcopoulos, Andrzej Lingas, Rasmus Pagh, and Takeshi Tokuyama. Efficiently correcting matrix products. *Algorithmica*, 79(2):428–443, 2017.
- 14 Jonathan I. Hall. Notes on coding theory. Available at <https://users.math.msu.edu/users/halljo/classes/codenotes/GRS.pdf>.
- 15 Wing-Kai Hon, Meng-Tsung Tsai, and Hung-Lung Wang. Verifying the product of generalized boolean matrix multiplication and its applications to detect small subgraphs. In *WADS*, 2023.
- 16 Mark A. Iwen, 2023. Personal Communication.
- 17 Mark A. Iwen and Craig V. Spencer. A note on compressed sensing and the complexity of matrix multiplication. *Information Processing Letters*, 109(10):468–471, 2009. [doi:10.1016/j.ipl.2009.01.010](https://doi.org/10.1016/j.ipl.2009.01.010).
- 18 Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *ICALP*, 2015.
- 19 Tracy Kimbrel and Rakesh Kumar Sinha. A probabilistic algorithm for verifying matrix products using  $O(n^2)$  time and  $\log_2 n + O(1)$  random bits. *Information Processing Letters*, 45(2):107–110, 1993.
- 20 Ivan Korec and Jiří Wiedermann. Deterministic verification of integer matrix multiplication in quadratic time. In *SOFSEM*, 2014.
- 21 Marvin Künnemann. On nondeterministic derandomization of Freivalds’ algorithm: Consequences, avenues and algorithmic progress. In *ESA*, 2018.
- 22 François Le Gall. Faster rectangular matrix multiplication by combination loss analysis. In *SODA*, 2024.
- 23 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *SODA*, 2018.
- 24 Ron M. Roth and Abraham Lempel. On MDS codes via Cauchy matrices. *IEEE Transactions on Information Theory*, 35(6):1314–1319, 1989.
- 25 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *SODA*, 2024.
- 26 Yu-Lun Wu and Hung-Lung Wang. Correcting matrix products over the ring of integers. *arxiv*, 2023. [arXiv:2307.12513](https://arxiv.org/abs/2307.12513).
- 27 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49(3):289–317, 2002.