A Simplified Reduction for Error Correcting Matrix Multiplication Algorithms

Igor Shinkar
Simon Fraser University, Burnaby, Canada

Harsimran Singh
Simon Fraser University, Burnaby, Canada

Abstract

We study the problem of transforming an algorithm for matrix multiplication, whose output has a small fraction of the entries correct into a matrix multiplication algorithm, whose output is fully correct for all inputs. In this work, we provide a new and simple way to transform an average-case algorithm that takes two matrices $A, B \in \mathbb{F}_p^{n \times n}$ for a prime p, and outputs a matrix that agrees with the matrix product AB on a $1/p + \epsilon$ fraction of entries on average for a small $\epsilon > 0$, into a worst-case algorithm that correctly computes the matrix product for all possible inputs.

Our reduction employs list-decodable codes to transform an average-case algorithm into an algorithm with one-sided error, which are known to admit efficient reductions from the work of Gola, Shinkar, and Singh [12]. Our reduction is more concise and straightforward compared to the recent work of Hirahara and Shimizu [18], and improves the overhead in the running time incurred during the reduction.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Matrix Multiplication, Reductions, Worst case to average case reductions

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.29

Category RANDOM

Acknowledgements We thank Valentine Kabanets for suggesting to extend the result of [12] to zero error model over arbitrary finite fields. We are also thankful to the anonymous reviewers for their valuable comments.

1 Introduction

The problem of efficiently multiplying two $n \times n$ matrices is one of the most fundamental problems in computer science. The naive approach for Matrix Multiplication takes $O(n^3)$ time, and there is a lot of ongoing research attempting to improve the running time. Strassen [29] first gave a well-known and widely adopted algorithm, with the running time of $O(n^{2.8074})$. Since then, a long line of work ([24, 5, 26, 25, 28, 8, 27, 31, 22, 3, 9, 32]) has achieved improvement in the exponent, with the current best known algorithm having the runtime of $O(n^{2.3713})$ [2]. It is still an open problem to find the optimal running time for Matrix Multiplication.

A natural and relaxed version of the above question is to come up with an algorithm which outputs the product of two matrices, but is allowed to have some incorrect entries. In this work, we address the problem of correcting the output of such an algorithm. Assume that we have an algorithm for matrix multiplication which gets two matrices and outputs a matrix which has a fraction of its entries correct, is it possible to correct the incorrect entries efficiently, i.e., without significantly increasing the running time? For two $n \times n$ matrices A and B, define their agreement, denoted by $\mathrm{agr}(A, B)$ as

$$\mathrm{agr}(A, B) = \frac{|\{(i, j) \mid A_{i,j} = B_{ij}\}|}{n^2}.$$

Hence, the problem can be stated as correcting the output of the matrix multiplication algorithm which has some agreement with the correct product. It is perhaps more convenient to view this question as the problem of getting a *worst-case to average-case approximate* reduction for matrix multiplication. Worst-case to Average-case reductions could help us to get fast algorithms which work in the worst-case, if we could come up with fast average-case algorithms. On the opposite side, such reductions allow us to show that if a problem is known to be worst-case hard, then it is also average-case hard. So our problem could be restated as
– Given an algorithm which gets two matrices $A, B \in \mathbb{F}_p^{n\times n}$, and outputs a matrix $C \in \mathbb{F}_p^{n\times n}$ such that C has a non-trivial agreement with AB in expectation, can we efficiently transform it into an algorithm which outputs the correct matrix product for all possible inputs?

1.1 Previous Work

Worst-Case to Average-case reductions for matrix multiplication have received considerable attention lately. Asadi, Golovnev, Gur, and Shinkar [4] and Hirahara and Shimizu [16] developed Worst-case to Average-case reductions for matrix multiplications. Specifically, they showed that if there exists an algorithm which, on input two matrices over a finite field, returns the correct matrix product on a small fraction of all possible inputs, then it is possible to transform it into an algorithm that computes the matrix product correctly on *all* the inputs.

Gola, Shinkar, and Singh [12] addressed the related problem of correcting matrix product efficiently, i.e., given an algorithm which on input two matrices over a finite field, outputs a matrix which agrees with the correct matrix product on a non-trivial fraction of entries, can it be reduced to an algorithm that computes the matrix product correctly on *all* the inputs? Note that this is a weaker assumption than the one used in previous works, since it only assumes a *fraction* of the output entries to be correct in expectation, whereas the previous works assumed that the output is *fully* correct for a small fraction of inputs. They showed that over a finite field when the output has a high agreement ($> 8/9$) with the correct matrix product, a reduction can be obtained using standard self-correction techniques for linear functions [6]. In the same work, using techniques from additive combinatorics, it was shown that over $\mathbb{F}_2$, an optimal reduction can be performed in the case of one-sided error, i.e., when all the 1 entries in the output are correct, and the expected agreement is $1/2 + \epsilon$, for any $\epsilon > 0$.

Hirahara and Shimizu [18] improved upon the previous results by providing a reduction from an algorithm which has a non-trivial agreement with the correct matrix product over a finite field (specifically, an agreement of $2/p + \epsilon$ for a field of size p), to a worst-case algorithm. Their key idea was to apply a left-right encoding to the output, using error correcting codes that are encodable and list-decodable in nearly linear time, allowing them to list-decode the left-right encoding efficiently, thereby correcting the matrix product with the help of a matrix-vector product verification algorithm to identify the correct item from the list. Over larger fields, Reed-Solomon Codes satisfied these requirements and allowed for optimal reductions. However, for small fields, they used Ta-Shma codes [30] which are based on random walks over expander graphs, and were shown to be nearly linear time list-decodable by Jeronimo, Srivastava and Tulsiani [21] and Jeronimo [20]. Using efficient list-decoding properties of Ta-Shma codes, a reduction (optimal upto a factor of 2) over small fields was obtained. By using the same codes and modifying the approach through the use of the notion of approximate list-decoding, an optimal reduction, i.e. from an agreement of $1/p + \epsilon$ for a field of size p, was obtained in [19].

1.2 Our Contributions

We provide a new worst-case to average-case approximate reduction over finite fields. Over the binary field $\mathbb{F}_2$, we prove the following.

▶ **Theorem 1.** *Let $n \in \mathbb{N}$ be sufficiently large, and let $\epsilon > 0$ be a small constant. Let* ALG *be an algorithm that gets as input two matrices $A, B \in \mathbb{F}_2^{n\times n}$ and outputs a matrix $\mathrm{ALG}(A,B) \in \mathbb{F}_2^{n\times n}$ in time $T(n)$ such that*

$$\mathop{\mathbb{E}}_{A,B\in\mathbb{F}_2^{n\times n}}[\mathrm{agr}(\mathrm{ALG}(A,B), A\cdot B)] \geq 1/2+\epsilon.$$

Then, there exists an algorithm ALG^* *running in time $2^{O(1/\epsilon^2)}\cdot\widetilde{O}(T(n))$ that gets as input two matrices $A, B \in \mathbb{F}_2^{n\times n}$ and outputs a matrix $\mathrm{ALG}^*(A,B) \in \mathbb{F}_2^{n\times n}$ such that* for all A, B

$$\Pr_{\mathrm{ALG}^*}[\mathrm{ALG}^*(A,B) = A\cdot B] > 1-o(1).$$

▶ Remark 2. Theorem 1 works for all $\epsilon \geq 1/(\log\log(n))^c$ for some absolute constant $c > 0$. The exact limitations on ϵ come from Theorem 15. If we construct a list decodable code that supports smaller ϵ (say, all $\epsilon > 1/\sqrt{\log(n)}$), then we can allow a smaller ϵ in the reduction in Theorem 1.

It is an open problem to show a reduction whose running time is $\mathrm{poly}(1/\epsilon)\cdot\widetilde{O}(T(n))$.

Note that any trivial algorithm which returns all 0 or all 1 will have an expected agreement of 1/2. Hence, this result is optimal in terms of agreement, since it allows us to reduce any non-trivial algorithm doing slightly better than a trivial algorithm. Similarly, over a finite field of prime order p, a worst-case to average-case reduction for an expected agreement of $1/p+\epsilon$ would be optimal.

▶ **Theorem 3.** *Let $n \in \mathbb{N}$ be sufficiently large, and let $\epsilon > 0$ be a small constant. Let* ALG *be an algorithm that gets as input two matrices $A, B \in \mathbb{F}_p^{n\times n}$ and outputs a matrix $\mathrm{ALG}(A,B) \in \mathbb{F}_p^{n\times n}$ in time $T(n)$ such that*

$$\mathop{\mathbb{E}}_{A,B\in\mathbb{F}_2^{n\times n}}[\mathrm{agr}(\mathrm{ALG}(A,B), A\cdot B)] \geq 1/p+\epsilon.$$

Then, there exists an algorithm ALG^* *running in time $2^{O(1/\epsilon^2)}\cdot p^{O(\log^2(1/\epsilon))}\cdot\widetilde{O}(T(n))$ that gets as input two matrices $A, B \in \mathbb{F}_p^{n\times n}$ and outputs a matrix $\mathrm{ALG}^*(A,B) \in \mathbb{F}_p^{n\times n}$ such that* for all A, B

$$\Pr_{\mathrm{ALG}^*}[\mathrm{ALG}^*(A,B) = A\cdot B] > 1-o(1).$$

The overhead in our reduction for a finite field of order p is exponential in $\mathrm{poly}(1/\epsilon)$. This improves upon the overhead of the reduction given by [18] over small fields, which is double-exponential in $\mathrm{poly}(1/\epsilon)$.

We believe that our reduction is conceptually simpler as compared to the reduction of [18]. For example, while both reductions use list decodable codes, in this paper we need the list decodable codes to have rather standard properties (which we construct in the Appendix), while [18] requires explicit expansion properties like splittablity of a collection of k-tuples. Our construction of the list-decodable code relies on simple code concatenation techniques, and does not require Walk-Amplified codes based on direct sum encodings of a collection of k-tuples obtained by taking random walks over expander graphs of large girth. In addition,

the code we construct has a full rank partition (Definition 6), which is a more natural notion than the (m, δ)-partition defined in [18]. Furthermore, we do not require the notions of left-right encoding and approximate list-decoding for obtaining optimal reductions.

Finally, in the process of obtaining an optimal reduction, we also provide a way to generalize the one-sided error reduction over the binary field [12] to any finite field of prime order.

2 Preliminaries

We begin by recalling some basic coding theory terminology. A code C over the Finite Field $\mathbb{F}_p$ is a subset $C \subseteq \mathbb{F}_p$. If C is a linear subspace of $\mathbb{F}_p$, it is called a linear code. When the field is $\mathbb{F}_2$, we call it a binary code. Elements of the code are referred to as codewords. A code can be equivalently described by an encoding, which is an injective mapping $\mathrm{Enc}\colon \mathbb{F}_p^n \to \mathbb{F}_p^N$, whose range gives the code C. For a linear code, the encoding $\mathrm{Enc}\colon \mathbb{F}_p^n \to \mathbb{F}_p^N$ can be expressed as $\mathrm{Enc}(x) = Gx$, where $G \in \mathbb{F}_p^{N\times n}$ is called a generator matrix. The rate of the code $C \subseteq \mathbb{F}_p^N$ is given by $\log_p(|C|)/N$, or equivalently as n/N. The distance of the code C is defined as $\min_{c_1,c_2\in C} \Delta(c_1, c_2)$, where $\Delta(c_1, c_2) = |\{i \mid (c_1)_i \neq (c_2)_i\}|/n$. [1] We refer the reader to [15] for further details on this subject.

A well-known linear code which we use in this work is the Reed-Solomon code.

▶ **Definition 4** (Reed-Solomon codes). *For $n, N \in \mathbb{N}$ and a prime $p \geq N$, a Reed-Solomon Encoding over a finite field $\mathbb{F}_p$ with the message $c = (c_0, \ldots, c_{n-1})$ and the set of evaluation points $\alpha = (\alpha_1, \ldots \alpha_N)$, denoted by $RS_{p,\alpha,n,N}\colon \mathbb{F}_p^n \to \mathbb{F}_p^N$, is given by $RS_{p,\alpha,n,N}(c) = (f(\alpha_1), \ldots, f(\alpha_N))$, where $f(x) = \sum_{i=0}^{n-1} c_i x^i$.*

We now define the notion of list-decoding.

▶ **Definition 5** (List-decodable codes). *A code $C \subseteq \mathbb{F}_p^N$ given by the encoding $\mathrm{Enc}\colon \mathbb{F}_p^n \to \mathbb{F}_p^N$ is (r, ℓ)-list-decodable if for all $y \in \mathbb{F}_p^N$, $|\{c \in C \mid \Delta(c, y) \leq r\}| \leq \ell$. A list-decoding algorithm is an algorithm which takes as input $y \in \mathbb{F}_p^N$ and with high probability, outputs all the vectors $x \in \mathbb{F}_p^n$ such that $\Delta(\mathrm{Enc}(x), y) \leq r$.*

Next we define the notion of *full-rank partition*. This notion is a special case of the notion of (m, δ) full-rank partition used in [16]. Our definition corresponds to $m = n$ and $\delta = 0$, which (in our opinion) is the most natural choice of parameters.

▶ **Definition 6** (Full-rank partition). *Let $N \geq n$ be integers such that $N = qn$ for some $q \in \mathbb{N}$. We say that a matrix $G \in \mathbb{F}_2^{N\times n}$ admits a* full-rank partition *if there is a partition $P_1, P_2, \ldots, P_q$ of $[N]$ such that:*

- *$|P_i| = n$ for all $i \in [q]$.*
- *For all $i \in [q]$, the rows indexed by P_i are linearly independent. Equivalently, the submatrix $G|_{P_i} \in \mathbb{F}_2^{n\times n}$ is full rank.*

We proceed to mention the linear list-decodable code which we crucially use in our reduction.

[1] Strictly speaking, it is called the relative distance of the code. However, we shall simply refer to it as distance in this work.

▶ **Lemma 7.** *Fix $\epsilon > 0$ and let $n \in \mathbb{N}$ be sufficiently large, such that $\epsilon > (1/\log\log(n))^c$ for some absolute constant $c > 0$. There exists a randomized algorithm that runs in time $O((\log\log(n)/\epsilon)^2)$ and with probability at least $2^{-O(1/\epsilon^2)}$ outputs an advice M_0, such that given M_0 we have the following.*

There exists a linear code $C\colon \mathbb{F}_2^n \to \mathbb{F}_2^N$, where $N/n = \mathrm{poly}(1/\epsilon)$ is an integer, and it satisfies the following properties.

1. *C is encodable in time $\mathrm{poly}(1/\epsilon) \cdot \widetilde{O}(n)$.*
2. *C is $(1/2 - \epsilon, \ell)$-list decodable for $\ell = \mathrm{poly}(1/\epsilon)$, and the running time of the decoder is $\mathrm{poly}(1/\epsilon) \cdot \widetilde{O}(n)$.*
3. *The generating matrix for C has a full-rank partition. Furthermore, there exists an algorithm that outputs the generating matrix and the partition in time $\mathrm{poly}(1/\epsilon) \cdot \widetilde{O}(n)$.*

We are certain that such codes are already known (see, e.g., [13]), however we could not find the exact statement in the literature. In particular, the notion of full-rank partition is a standard notion studied in the literature. We refer the reader to Section A for the proof of Lemma 7.

We now state a variant of Markov's inequality which we use repeatedly in this work.

▶ **Lemma 8** (Markov's Inequality)**.** *Let X be a random variable taking values in the interval $[0, 1]$ that has non-zero mean μ. Then, for any $\lambda \in [0, \mu]$,*

$$\Pr[X \geq \mu - \lambda] \geq \frac{\lambda}{1 - \mu + \lambda}.$$

Proof. Let $p = \Pr[X \geq \mu - \lambda]$. Then,

$$\begin{aligned}\mu &= \sum_x x \Pr[X = x] \\ &\leq (\mu - \lambda) \cdot \Pr[X < \mu - \lambda] + 1 \cdot \Pr[X \geq \mu - \lambda] \\ &= (\mu - \lambda) \cdot (1 - p) + p\end{aligned}$$

This gives us that $p \geq \lambda/(1 - \mu + \lambda)$. ◀

We now state the verification algorithms we use in the reduction. Firstly, the well-known Freivald's Randomized Algorithm for verifying matrix multiplication. Secondly, the data structure algorithm given by Hirahara and Shimizu [16] for verifying matrix-vector multiplication.

▶ **Lemma 9** (Freivald's Algorithm [11])**.** *There exists a randomized algorithm R with runtime $O(kn^2)$ which, when given as input three matrices $A, B, C \in \mathbb{F}_p^{n \times n}$, outputs either YES or NO such that:*

- $\Pr_R[R(A, B, C) = YES \mid A \cdot B = C] = 1$.
- $\Pr_R[R(A, B, C) = NO \mid A \cdot B \neq C] \geq 1 - 2^{-k}$.

▶ **Lemma 10** (Lemma 6.2 from [17])**.** *There exists a data structure algorithm $D = (D_{pre}, D_{query})$ such that:*

- *$D_{pre}(M)$ is a randomized algorithm which gets as input $M \in \mathbb{F}_p^{m \times n}$, and outputs a string σ in $O(mn \log m)$ time.*
- *$D^{\sigma}_{query}(x, y)$ is a deterministic oracle algorithm which, when given oracle access to σ and inputs $x, y \in \mathbb{F}_p^m$, verifies whether $x = My$ in $O(m \log m)$ time, and is correct with probability $1 - 1/\mathrm{poly}(m)$ over the internal coin tosses of D_{pre}.*

Finally, we mention the one-sided error result from [12] which we use in our reduction.

▶ **Lemma 11** (Theorem 3 from [12]). *Let $\delta > 0$ be a constant. If there exists an algorithm* ALG *running in time $T(n)$ such that*

- $\Pr_{\substack{A,B\in\mathbb{F}_2^{n\times n}\\ i,j\in[n]}}[\mathrm{ALG}(A,B)_{(i,j)} = 1] \geq \delta$
- *If $(AB)_{i,j} = 0$, then $\mathrm{ALG}(A,B)_{i,j} = 0$.*

Then, there exists an algorithm $\widetilde{\mathrm{ALG}}$ with runtime $\widetilde{T}(n) = 2^{O(\log^3(1/\delta))} \cdot O(\log n) \cdot T(n)$, such that for all $A, B \in \mathbb{F}_2^{n\times n}$,

$$\Pr_{\widetilde{\mathrm{ALG}}}[\widetilde{\mathrm{ALG}}(A,B) = A \cdot B] > 1 - o(1).$$

3 Reduction to the one-sided error case

In this section, we present a reduction of matrix multiplication over the binary field to the one-sided error case of Lemma 11, using a binary linear code whose generating matrix has a full-rank partition. Specifically, we prove the following lemma.

▶ **Lemma 12.** *For some constant $\epsilon > 0$, let $BLC\colon \mathbb{F}_2^n \to \mathbb{F}_2^N$ be a binary linear code which can be encoded in time T_{enc} and is ℓ-list decodable within radius $1/2 - \epsilon/4$ in time T_{dec}. Furthermore, assume that the generating matrix of BLC has a full-rank partition, such that the generating matrix and the partition are computable in time T_{par}.*

Suppose there exists an algorithm ALG *running in time $T(n)$ such that for inputs $A, B \in \mathbb{F}_2^{n\times n}$*

$$\mathop{\mathbb{E}}_{A,B\in\mathbb{F}_2^{n\times n}}[\mathrm{agr}(\mathrm{ALG}(A,B), A\cdot B)] \geq 1/2 + \epsilon.$$

Then, there exists an algorithm ALG^*, such that for inputs $A, B \in \mathbb{F}_2^{n\times n}$ it holds that

- *If $(AB)_{(i,j)} = 0$, then $\mathrm{ALG}^*(A,B)_{(i,j)} = 0$, and*
- $\Pr_{\substack{A,B\\ i,j}}[\mathrm{ALG}^*(A,B)_{(i,j)} = 1] \geq \epsilon^2/8$,

and the running time of ALG^* is $T'(n) = n\cdot T_{enc} + T_{par} + O(\frac{N}{n})\cdot T(n) + O(Nn) + O(n^2\log n) + O(n\cdot(T_{dec} + \ell\cdot n\log n))$.

The reduction is outlined in Algorithm 1. We now describe the steps of the reduction.

Let $G \in \mathbb{F}_2^{N\times n}$ be the generator matrix representing the binary linear code $BLC\colon \mathbb{F}_2^n \to \mathbb{F}_2^N$. For a matrix $M \in \mathbb{F}_2^{n\times n}$, define $\mathrm{Enc}(M) = MG^T$, i.e. *Enc* is a function $\mathrm{Enc}\colon \mathbb{F}_2^{n\times n} \to \mathbb{F}_2^{n\times N}$ applying the binary linear code BLC to every row of M.

Since G has a full-rank partition, its rows can be partitioned into sets $P_1, P_2, \ldots P_t$, such that for every i between 1 and t, $|P_i| = n$ and the sub matrix $G|_{P_i}$ is full rank.

Now, for $A, B \in \mathbb{F}_2^{n\times n}$, we construct a matrix $C \in \mathbb{F}_2^{n\times N}$ such that $\mathrm{agr}(C, \mathrm{Enc}(AB)) \geq 1/2 + \epsilon$ in expectation. To this end, we note that if $B \in \mathbb{F}_2^{n\times n}$ is chosen uniformly at random, then the matrix given by $BG^T|_{P_i} \in \mathbb{F}_2^{n\times m}$ will also be a uniformly random matrix, since $G^T|_{P_i}$ is full-rank. Our idea is to apply the encoding given by the sub-matrices $G^T|_{P_i}$ and then constructing C by concatenating the columns of all the resultant matrices. Formally, we write C as:

$$C = ALG(A, B\cdot G^T|_{P_1}) \circ \cdots \circ ALG(A, B\cdot G^T|_{P_t}) = ALG(A, BG^T),$$

Algorithm 1 Reduction to the one-sided error case.

Input: $A, B \in \mathbb{F}_2^{n\times n}$, ALG, The generating matrix $G \in \mathbb{F}_2^{N\times n}$ of the code BLC.
Output: An $n \times n$ matrix $M \in \mathbb{F}_2^{n\times n}$

Find the full-rank partition $P_1, P_2, \ldots, P_t \in [N]$ of the rows of G.
Initialize the matrix $M \in \mathbb{F}_2^{n\times n}$ to all zeros.
for $i \in [t]$ **do**
    Define $C_i = \mathrm{ALG}(A, B \cdot G^T|_{P_i})$
Construct C by concatenating $C_1, C_2, \ldots, C_t$ column wise.
Run D_{pre} from Lemma 10 with input B^T.
for $i \in [n]$ **do**
    For the i^{th} row vector c_i^T of C, run the list-decoding algorithm from Theorem 15 with input c_i, to get a set of vectors $S \subseteq \mathbb{F}_2^n$ such that $|S| \leq l$.
    For every $s \in S$, run $D_{query}(s, A_i)$ from Lemma 10, where A_i^T is the i^{th} row vector of A. If it outputs TRUE, then replace the i^{th} row vector of M with s^T.
return M.

where $\circ$ denotes the column concatenation. Now, we observe that

$$\begin{aligned}\mathop{\mathbb{E}}_{A,B\in\mathbb{F}_2^{n\times n}}[\mathrm{agr}(ABG^T, C)] &= \mathop{\mathbb{E}}_{i\in[t]}\left[\mathop{\mathbb{E}}_{A,B\in\mathbb{F}_2^{n\times n}}\left[\mathrm{agr}(\mathrm{ALG}(A, B\cdot G^T|_{P_i}), ABG^T|_{P_i})\right]\right]\\ &\geq 1/2+\epsilon,\end{aligned}$$

where the last inequality follows from the fact that for every $1 \leq i \leq t$, $B \cdot G^T|_{P_i}$ is a uniformly random matrix. In other words, all the columns are generated by taking the product of two uniformly random matrices, for which ALG has an agreement of $1/2 + \epsilon$.

Since $\mathbb{E}_{A,B}[\mathrm{agr}(C, \mathrm{Enc}(AB))] > 1/2 + \epsilon$, by Markov's inequality (Lemma 8), there must exist at least ϵ fraction of rows of C such that their agreement with the corresponding rows of $Enc(AB)$ is at least $1/2 + \epsilon/2$ in expectation. Formally, there is a set $R \subseteq [n]$ of rows of size $|R| \geq \epsilon n$ such that $\mathbb{E}_{A,B}[\mathrm{agr}(c_i, BLC(d_i))] > 1/2 + \epsilon/2$ for all $i \in R$, where c_i^T is the row indexed by i in C and d_i^T is the row indexed by i in AB.

Now, for each row $i \in R$, by another application of Markov's inequality (Lemma 8) we can imply that there exists a subset of good inputs $GOOD_i$ of density at least $\epsilon/2$ such that if $(A, B) \in GOOD_i$, $\mathrm{agr}(c_i, BLC(d_i))] > 1/2 + \epsilon/4$. Therefore, with probability at least $\epsilon \cdot \epsilon/2$, the distance of the row i of C from the encoding of the row i of the correct product AB is at most $1/2 - \epsilon/4$.

Hence, for random choices of A and B, we can list decode these rows in nearly linear time and identify the correct row using an efficient Matrix-Vector product verification algorithm. Furthermore, the verification algorithm (Lemma 10) also lets us identify these good rows.

The Matrix-Vector product verification algorithm gets each vector $s \in \mathbb{F}_2^n$ from the list obtained by decoding the row i, and checks whether $s = B^T A_i$, where A_i^T is the vector representing the row indexed by i in A. If the output is YES, it implies that s^T is the row i of the correct matrix product AB.

Finally, fixing all entries in the undecoded rows by 0 takes us to the one-sided error setting of [12], as stated in Lemma 11. That is,

- If $(AB)_{(i,j)} = 0$, then $M_{(i,j)} = 0$, and
- For at least $\epsilon/2$ fraction of inputs $A, B \in \mathbb{F}_2^{n\times n}$, there are ϵn rows $R \subseteq [n]$ such that $(AB)_{(i,j)} = M_{(i,j)}$ holds for all $(i, j) \in R \times [n]$.

This implies the following.

▷ Claim 13. For random $A, B \in \mathbb{F}_2^{n\times n}$ and M obtained by applying *Algorithm* 1 we have the following

- If $(AB)_{(i,j)} = 0$, then $M_{(i,j)} = 0$, and
- $\Pr_{\substack{A,B\in\mathbb{F}_2^{n\times n}\\ i,j\in[n]}}[M_{(i,j)} = 1] \geq \epsilon^2/8$.

Proof. The first item follows directly from the reduction. For the second item, observe that the reduction guarantees that $\Pr[M_{(i,j)} = (AB)_{(i,j)}] \geq \epsilon^2/2$, since for ϵ fraction of rows i we have at least $\epsilon/2$ fraction of the inputs (A, B) such that M agree with $A \cdot B$ on the i'th row.

For item 2, note that if for each $i \in R$ we had the i'th row of M correct *for all* A, B, then we would have $\Pr[M_{(i,j)} = 1] \geq \epsilon^2/4 - 2^{-n}$, since $(AB)_{(i,j)}$ is equally likely to be 0 or 1 for any i, j unless the i'th row of A is all zeros, which happens with probability 2^{-n}.

However, by the argument above, for each $i \in R$ we have only $\epsilon/2$ fraction of the inputs (A, B) for which the i'th row of M is correct. By the standard concentration inequality, the fraction of ones in each such row is at least $\epsilon^2/6$ for all except for $2^{-\Omega(n)}$ fraction of the inputs, and thus $\Pr[M_{(i,j)} = 1] \geq \epsilon^2/6 - 2^{-\Omega(n)} \geq \epsilon^2/8$, as required. ◁

Time Complexity of Algorithm 1

Encoding the n rows of B can be done in time $n \cdot T_{enc}$ and partitioning the rows of G can be done in time T_{par}. Steps 3 and 4 can be performed in $O(\frac{N}{n} \cdot T(n))$ time. Step 5 takes $O(Nn)$ time. From Lemma 10, step 6 takes $O(n^2 \log n)$ time and Steps 7 to 9 can be performed in time $O(n \cdot (T_{dec} + \ell \cdot n \log n))$. The rest of the steps take $O(n^2)$ time.

This completes the proof of Lemma 12.

4 Worst-Case to Average-Case reduction

We now use Lemma 12 and Lemma 11 to obtain a worst case to average case reduction. In particular, we provide the proof of Theorem 1.

▶ **Theorem 1.** *Let $n \in \mathbb{N}$ be sufficiently large, and let $\epsilon > 0$ be a small constant. Let* ALG *be an algorithm that gets as input two matrices $A, B \in \mathbb{F}_2^{n\times n}$ and outputs a matrix* $\mathrm{ALG}(A, B) \in \mathbb{F}_2^{n\times n}$ *in time $T(n)$ such that*

$$\mathop{\mathbb{E}}_{A,B\in\mathbb{F}_2^{n\times n}}[\mathrm{agr}(\mathrm{ALG}(A, B), A \cdot B)] \geq 1/2 + \epsilon.$$

Then, there exists an algorithm ALG^* *running in time $2^{O(1/\epsilon^2)} \cdot \widetilde{O}(T(n))$ that gets as input two matrices $A, B \in \mathbb{F}_2^{n\times n}$ and outputs a matrix* $\mathrm{ALG}^*(A, B) \in \mathbb{F}_2^{n\times n}$ *such that* for all A, B

$$\Pr_{\mathrm{ALG}^*}[\mathrm{ALG}^*(A, B) = A \cdot B] > 1 - o(1).$$

Proof. We start by applying the reduction from Lemma 12 with the list-decodable code from Lemma 7. Assuming that the code from Lemma 7 satisfies the required properties, which happens with probability at least $2^{-O(1/\epsilon^2)}$, we get an algorithm ALG^* whose running time is $T'(n) = \mathrm{poly}(1/\epsilon)\widetilde{O}(n^2) + \mathrm{poly}(1/\epsilon) \cdot T(n)$ and it satisfies the conditions of Lemma 11 with $\delta = \epsilon^2/8$.

Then, by applying Lemma 11, we get an algorithm $\widetilde{\text{ALG}}$ whose running time is $\widetilde{T}(n) = 2^{O(\log^3(1/\epsilon))} \cdot O(\log n) \cdot T'(n)$, and it solves the matrix multiplication problem in worst case with high probability.

Finally, we can run $\widetilde{\text{ALG}}$, and verify the resulting product using Freivald's Matrix Multiplication Algorithm (Lemma 9). Since the code in Lemma 7 satisfies the required properties with probability at least $(0.288)^{O(1/\epsilon^2)}$, we can repeat the procedure $2^{O(1/\epsilon^2)}$ times to get the correct output with high probability. Therefore, the total running time of the algorithm we get is $2^{O(1/\epsilon^2)} \cdot \widetilde{T}(n)$, and since $T(n) \geq n^2$, it implies that the total running time is $2^{O(1/\epsilon^2)} \cdot \widetilde{O}(T(n))$. ◀

5 Reduction for a Finite Field of Prime order

In this section, we show that our reduction can be generalized to any finite field $\mathbb{F}_p$, where p is a prime.

This can be achieved by extending the one-sided error model of [12] to a zero-error model over $\mathbb{F}_p$. In the one sided error condition of Lemma 11, the output 1 can be viewed as corresponding to the correct entry, while 0 can be viewed as corresponding to $\perp$ or "Don't know".

The proof of Lemma 11 relied on the techniques from Additive Combinatorics. The reduction given by [12] for the one-sided error model proceeded by defining the good coordinates, which output 1 with a significant probability on random input matrices. Thereafter, they expressed the input matrices as the sum of random matrices, and used the Probabilistic Bogolyubov Lemma to retrieve all the 1 entries in the correct output. Although they proved the Probabilistic Bogolyobuv lemma for the binary field $\mathbb{F}_2$, their proof can be extended through similar techniques for Fourier analysis over F_p. Hence, the worst-case algorithm ALG^* in Lemma 14 can be obtained in the same way as the reduction for the one-sided error model, by tweaking the definition of good coordinates to be the coordinates which are not $\perp$ with a significant probability.

Hence, the lemma can be generalized as follows.

▶ **Lemma 14.** *Let $\delta > 0$ be a constant and p be a prime. If there exists an algorithm* ALG *running in time $T(n)$ such that for all $A, B \in \mathbb{F}_p^{n\times n}$*

- $\Pr_{\substack{A,B\in\mathbb{F}_p^{n\times n}\\ i,j\in[n]}}[\text{ALG}(A,B)_{(i,j)} = (AB)_{(i,j)}] \geq \delta$
- $\Pr_{\substack{A,B\in\mathbb{F}_p^{n\times n}\\ i,j\in[n]}}[\text{ALG}(A,B)_{i,j} \notin \{(AB)_{i,j}, \perp\})] = 0.$

Then, there exists an algorithm $\widetilde{\text{ALG}}$ running in time $\widetilde{T}(n) = 2^{O(\log^3(1/\delta))} \cdot p^{O(\log^2(1/\delta))} \cdot O(\log n) \cdot T(n)$ such that for all $A, B \in \mathbb{F}_p^{n\times n}$ it holds that

$$\Pr_{\widetilde{\text{ALG}}}[\widetilde{\text{ALG}}(A,B) = A \cdot B] > 1 - o(1).$$

Below we sketch the proof of the lemma, mostly referencing the reduction analyzed in [12].

Proof. The lemma follows by slightly tweaking the reduction for the one-sided error regime, given in [12]. Similar to [12, Definition 12], we define the set of good coordinates G to be

$$G = \{(i,j) \in [n] \times [n] : \Pr_{A,B\in\mathbb{F}_p^{n\times n}}[\text{ALG}(A,B)_{i,j} \neq \perp] > \delta/2\}.$$

Using the algorithm for approximating good coordinates [12, Algorithm 2] and modifying the if-condition in line 12 to take into account the new definition of good coordinates G, we get a zero-error algorithm Z such that for $(i,j) \in G$, $\Pr[Z(A,B)_{i,j} = (A \cdot B)_{i,j}] \geq 2^{-O(\log^3(1/\delta))} \cdot p^{-O(\log^2(1/\delta))}$ [12, Claim 19]. This is the most technical step in the proof, relying on the Probabilistic Bogolyubov Lemma. Finally, using random permutations and running Z several times [12, Algorithm 3], we get the desired result. ◀

For generalizing Theorem 1 to $\mathbb{F}_p$, we can proceed in the same way as in Algorithm 1 by initializing all the entries of M to be $\perp$ rather than zero. This enables us to get a reduction to the zero-error case of Lemma 14. For this reduction, we need a linear code over $\mathbb{F}_p$ which is list-decodable from a radius of $1/p - \epsilon/2$ and admits a full-rank partition. This can be achieved by generalizing the construction of the binary linear code in Lemma 7 to a linear code with alphabet size p, which is addressed in Remark 20 in the appendix.

Hence, we get the following generalization for a field of prime order.

▶ **Theorem 3.** *Let $n \in \mathbb{N}$ be sufficiently large, and let $\epsilon > 0$ be a small constant. Let* ALG *be an algorithm that gets as input two matrices $A, B \in \mathbb{F}_p^{n \times n}$ and outputs a matrix $\mathrm{ALG}(A,B) \in \mathbb{F}_p^{n \times n}$ in time $T(n)$ such that*

$$\mathop{\mathbb{E}}_{A,B \in \mathbb{F}_2^{n \times n}} [\mathrm{agr}(\mathrm{ALG}(A,B), A \cdot B)] \geq 1/p + \epsilon.$$

Then, there exists an algorithm ALG^* *running in time $2^{O(1/\epsilon^2)} \cdot p^{O(\log^2(1/\epsilon))} \cdot \widetilde{O}(T(n))$ that gets as input two matrices $A, B \in \mathbb{F}_p^{n \times n}$ and outputs a matrix $\mathrm{ALG}^*(A,B) \in \mathbb{F}_p^{n \times n}$ such that* for all A, B

$$\Pr_{\mathrm{ALG}^*}[\mathrm{ALG}^*(A,B) = A \cdot B] > 1 - o(1).$$

References

1. M. Alekhnovich. Linear diophantine equations over polynomials and soft decoding of reed-solomon codes. *IEEE Transactions on Information Theory*, 51(7):2257–2265, 2005. doi:10.1109/TIT.2005.850097.
2. Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039. SIAM, 2025. doi:10.1137/1.9781611978322.63.
3. Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA 2021*, pages 522–539. SIAM, 2021. doi:10.1137/1.9781611976465.32.
4. Vahid R. Asadi, Alexander Golovnev, Tom Gur, and Igor Shinkar. Worst-case to average-case reductions via additive combinatorics. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 1566–1574, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520041.
5. Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti. O(n2.7799) complexity for n × n approximate matrix multiplication. *Information Processing Letters*, 8(5):234–235, 1979. doi:10.1016/0020-0190(79)90113-3.
6. Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.
7. A. Borodin and R. Moenck. Fast modular transforms. *Journal of Computer and System Sciences*, 8(3):366–386, 1974. doi:10.1016/S0022-0000(74)80029-2.

8 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. Computational algebraic complexity editorial. doi:10.1016/S0747-7171(08)80013-2.

9 Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 2129–2138. IEEE, 2023. doi:10.1109/FOCS57990.2023.00130.

10 Charles M. Fiduccia. Polynomial evaluation via the division algorithm the fast fourier transform revisited. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, STOC '72, pages 88–93, New York, NY, USA, 1972. Association for Computing Machinery. doi:10.1145/800152.804900.

11 Rusins Freivalds. Probabilistic machines can use less running time. In *IFIP congress*, volume 839, page 842, 1977.

12 Ashish Gola, Igor Shinkar, and Harsimran Singh. Matrix Multiplication Reductions. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024)*, volume 317 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2024.34.

13 Venkatesan Guruswami. List decoding of binary codes–a brief survey of some recent results. In Yeow Meng Chee, Chao Li, San Ling, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, pages 97–106, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-01877-0_10.

14 Venkatesan Guruswami, Johan Håstad, and Swastik Kopparty. On the list-decodability of random linear codes. *IEEE Transactions on Information Theory*, 57(2):718–725, 2011. doi:10.1109/TIT.2010.2095170.

15 Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory, 2023. Draft available at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/.

16 Shuichi Hirahara and Nobutaka Shimizu. Hardness self-amplification: Simplified, optimized, and unified. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 70–83, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3564246.3585189.

17 Shuichi Hirahara and Nobutaka Shimizu. Hardness self-amplification: Simplified, optimized, and unified, 2023. Full version of [16]. URL: https://eccc.weizmann.ac.il/report/2023/026/.

18 Shuichi Hirahara and Nobutaka Shimizu. Error-correction of matrix multiplication algorithms. In *Proceedings of the 57th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2025. Association for Computing Machinery, 2025.

19 Shuichi Hirahara and Nobutaka Shimizu. An optimal error-correcting reduction for matrix multiplication. In *Proceedings of the 52nd EATCS International Colloquium on Automata, Languages, and Programming*, ICALP, pages 97:1–97:17, 2025. doi:10.4230/LIPIcs.ICALP.2025.97.

20 Fernando Granha Jeronimo. Fast Decoding of Explicit Almost Optimal ϵ-Balanced q-Ary Codes And Fast Approximation of Expanding k-CSPs. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 60:1–60:16, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.60.

21 Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani. Near-linear time decoding of ta-shma's codes via splittable regularity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1527–1536, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451126.

22 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2608628.2608664.

23 Ray Li and Mary Wootters. Improved List-Decodability of Random Linear Binary Codes. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:19, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.50.

24 V. Ya. Pan. Strassen's algorithm is not optimal trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations. In *19th Annual Symposium on Foundations of Computer Science (SFCS 1978)*, pages 166–176, 1978. doi:10.1109/SFCS.1978.34.

25 Francesco Romani. Some properties of disjoint sums of tensors related to matrix multiplication. *SIAM Journal on Computing*, 11(2):263–267, 1982. doi:10.1137/0211020.

26 A. Schönhage. Partial and total matrix multiplication. *SIAM Journal on Computing*, 10(3):434–455, 1981. doi:10.1137/0210032.

27 Andrew James Stothers. On the complexity of matrix multiplication, 2010. URL: https://api.semanticscholar.org/CorpusID:262795811.

28 V. Strassen. The asymptotic spectrum of tensors and the exponent of matrix multiplication. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 49–54, 1986. doi:10.1109/SFCS.1986.52.

29 Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

30 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 238–251, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3055399.3055408.

31 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2213977.2214056.

32 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3792–3835. SIAM, 2024. doi:10.1137/1.9781611977912.134.

A List decodable codes

▶ **Theorem 15.** *Fix $\epsilon > 0$ and let $n \in \mathbb{N}$ be sufficiently large, such that $\epsilon > (1/\log\log(n))^c$ for some absolute constant $c > 0$. There exists a randomized algorithm that runs in time $O((\log\log(n)/\epsilon)^2)$ and with probability at least $2^{-O(1/\epsilon^2)}$ outputs an advice M_0, such that given M_0 we have the following.*

There exists a linear code $C\colon \mathbb{F}_2^n \to \mathbb{F}_2^N$, where $N/n = \mathrm{poly}(1/\epsilon)$ is an integer, and it satisfies the following properties.

1. *C is encodable in time $\mathrm{poly}(1/\epsilon) \cdot \widetilde{O}(n)$.*
2. *C is $(1/2 - \epsilon, \ell)$-list decodable for $\ell = \mathrm{poly}(1/\epsilon)$, and the running time of the decoder is $\mathrm{poly}(1/\epsilon) \cdot \widetilde{O}(n)$.*
3. *The generating matrix for C has a full-rank partition, and there exists an algorithm that outputs the partition in time $\mathrm{poly}(1/\epsilon) \cdot \widetilde{O}(n)$.*

▶ Remark 16. We make two comments about the advice M_0 in Theorem 15.

1. The code C in Theorem 15 depends on M_0, and the running times of the algorithms for encoding and list-decoding are $\text{poly}(1/\epsilon) \cdot \widetilde{O}(n)$ when given the advice M_0.
2. Given M_0 we can verify that it satisfies the desired properties in time $\log(n)^{O(1/\epsilon^2)}$. In particular, by sampling $2^{O(1/\epsilon^2)}$ independent advices and verifying them, we can find the advice M_0 with the desired properties in time $\log(n)^{O(1/\epsilon^2)}$ with probability 0.99.

Our construction will use the following result on list decoding of Reed-Solomon codes.

▶ **Lemma 17** (List Decoding Reed-Solomon Codes [1]). *Let $\mathbb{F}_q$ be a finite field, and let $\alpha = (\alpha_1, \dots \alpha_N) \subseteq \mathbb{F}_q^N$. Consider the Reed-Solomon code $RS_{q,\alpha,n,N} \colon \mathbb{F}_q^n \to \mathbb{F}_q^N$ of degree n. For $\delta > \sqrt{n/N}$ the code $RS_{q,\alpha,n,N}$ is $(1-\delta, O(1/\delta))$-list decodable, and the list-decoding algorithm runs in time $(N/n)^{O(1)} \cdot O(N \log^2(N))$*

We start the proof of Theorem 15 with the following simple claim.

▷ Claim 18. Fix $0 < \delta < 1/2$, and let $K = \lceil 3/\delta^2 \rceil$. For a sufficiently large $t \in \mathbb{N}$ such that $t > \text{poly}(1/\delta)$ there exists a linear code $C_0 \colon \mathbb{F}_2^t \to \mathbb{F}_2^{Kt}$ defined as $C_0(x) = Mx$ for some generating matrix $M \in \mathbb{F}_2^{Kt \times t}$ such that

- C_0 is $(1/2 - \delta, 2/\delta^2)$-list decodable, and
- M has a full-rank partition defined as $(\{1, \dots, t\}, \{t+1, \dots, 2t\}, \dots \{(K-1)t+1, \dots Kt\})$.

A uniformly random matrix $M \in \mathbb{F}_2^{Kt \times t}$ satisfies the desired properties with probability at least $0.288^K - 2^{-\text{poly}(\delta)t}$. Furthermore, given a matrix M, we can verify that it satisfies the desired properties in time $2^{O(Kt)}$.

Proof. Let $M \in \mathbb{F}_2^{Kt \times t}$ be a uniformly random matrix. Let $C_0 = \{Mx : x \in \mathbb{F}_2^t\} \subseteq \mathbb{F}^{Kt}$ be the linear code corresponding to the generating matrix M.

By the result of Li and Wootters [23, Theorem 5], with probability $1 - 2^{-\text{poly}(\delta)t}$ the code C is $(1/2 - \delta, 2/\delta^2)$-list decodable.

Next we show that with probability at least 0.288^K the random matrix M has the full-rank partition $(\{1, \dots, t\}, \{t+1, \dots, 2t\}, \dots \{(K-1)t+1, \dots Kt\})$. Note that in each block of t rows are linearly independent with probability

$$\prod_{i=0}^{t-1} \left(1 - \frac{2^i}{2^t}\right) \geq \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{7}{8} \cdot \frac{15}{16} \cdots > 0.288788095.$$

By independence, between blocks we get that M has a full-rank partition with probability at least 0.2887^K. Since t is sufficiently large, both events happen with probability at least $0.2887^K - 2^{-\text{poly}(\delta)t} > 0.288^K$.

For the "furthermore" statement of the claim, note that if we are given a matrix M, then we can check that for all possible strings $w \in \mathbb{F}_2^{Kt}$ there are at most $2/\delta^2$ codewords at distance $(1/2 - \delta)$ in total time $2^{Kt} \times 2^t \times \text{poly}(Kt)$, as required. ◁

Next we prove Theorem 15 by concatenating Reed-Solomon code as the outer code with the code from Claim 18 as the inner code. Specifically, we use the following lemma.

▶ **Lemma 19** (Concatenating codes). *Suppose $C_{in} \colon \mathbb{F}_2^t \to \mathbb{F}_2^{K_{in}t}$ is a linear code defined as $C_{in}(x) = Mx$ for some matrix $M \in \mathbb{F}_2^{K_{in}t \times t}$ such that*

- *C_{in} has an encoding algorithm whose running time is $E(t)$,*
- *C_{in} is $(1/2 - \epsilon/2, \ell)$-list decodable, and the running time of the list decoding algorithm is $D(t)$,*

- *C_{in} has a full-rank partition defined as $(\{1,\dots,t\},\{t+1,\dots,2t\},\dots\{(K-1)t+1,\dots Kt\})$. Let $4(\ell/\epsilon)^2 \le K_{out} \le 8(\ell/\epsilon)^2$ be a power of 2.*

Then, there exists a linear code $C\colon \mathbb{F}_2^n \to \mathbb{F}_2^{Kn}$ with $n = t\cdot 2^t/K_{out}$ and $K = K_{out}\cdot K_{in}$, such that

1. *There is an encoding algorithm for C with running time $\tilde{O}(Kn\cdot E(t))$,*
2. *C is $(1/2-\epsilon, O(\ell/\epsilon))$-list decodable,*
3. *The list decoding algorithm runs in time $D(t)\cdot 2^t + \mathrm{poly}(\ell/\epsilon)\cdot\widetilde{O}(2^t)$,*
4. *C has a full-rank partition.*

Proof. We define C as the concatenation of Reed-Solomon code as the outer code and C_{in} as the inner code. Specifically, the encoding of C works as follows.

Given a message x of length $n = t\cdot 2^t/K_{out}$ view it as $2^t/K_{out}$ blocks of length t each. Treating each block as an element of the finite field $\mathbb{F}_q$ with $q = 2^t$ we first apply the Reed-Solomon code $RS\colon q^{2^t/K_{out}} \to q^{2^t}$ on $x \in \mathbb{F}_q^{2^t/K_{out}}$ to get an encoding in $\mathbb{F}_q^{2^t}$. Then, we apply the code $C_{in}\colon \mathbb{F}_2^t \to \mathbb{F}_2^{K_{in}t}$ to each symbol of the Reed-Solomon codeword. Therefore, C is a linear code that takes $n = t\cdot 2^t/K_{out}$ bits and outputs $K_{in}\cdot K_{out}\cdot n$ bits.

Next we prove that C satisfies the properties stated in the lemma.

1. Using the fast multipoint evaluation of polynomials [10, 7], Reed-Solomon codes are encodable in $\widetilde{O}(2^t)$ time. Given the Reed-Solomon encoding, we apply C_0 on each block, which takes $E(t)$ time for each of the 2^t blocks. Therefore, the total running time is dominated by $\widetilde{O}(2^t\cdot E(t))$.
2. For the decoding algorithm, suppose we get a word $w \in \mathbb{F}_2^n$ that is $1/2-\epsilon$ close to an encoding of some message $c^* = C(x^*)$. By Markov's inequality there are at least ϵ fraction of blocks that are $1/2-\epsilon/2$ close to C_{in}.

   We run the $(1/2-\epsilon/2,\ell)$-list decoding algorithm for C_{in} on each block of w, and get a list of size at most ℓ for each block. By choosing a uniformly random element in each list, we obtain a word of length 2^t over the alphabet $\mathbb{F}_q$ that agrees with the Reed-Solomon encoding of x in at least ϵ/ℓ fraction of symbols in expectation. Furthermore, by Chernoff bound, with probability $1-e^{-\Omega(\epsilon\cdot 2^t/\ell)}$ the word we obtain agrees with the Reed-Solomon encoding of x in at least $\epsilon/2\ell$ fraction of symbols. Let us assume from now on the event that at least $\epsilon/2\ell$ fraction of symbols are correct. Since $\epsilon/2\ell \ge \sqrt{1/K_{out}}$, we can apply Lemma 17 to obtain a list of length at most $O(\ell/\epsilon)$ that contains our message x^*, as required.

   Therefore, any string $x \in \mathbb{F}_2^n$ belongs to the list of outputs with probability at least $1-e^{-\Omega(\epsilon\cdot 2^t/\ell)}$. Therefore, by union bound the output of the decoding algorithm contains all strings with probability at least $1-O(\ell/\epsilon)\cdot e^{-\Omega(\epsilon\cdot 2^t/\ell)} > 1-e^{-\Omega(\epsilon\cdot 2^t/\ell)}$, where the $\Omega(\cdot)$ notations might hide different constants.
3. For the running time of the list decoder, we first run the decoder for C_{in} on each of the 2^t blocks, which take total $2^t\cdot D(t)$ time. After that we sample one element from each list, and run the list decoder for Reed-Solomon, which runs in time $\mathrm{poly}(K_{out})\cdot\widetilde{O}(2^t)$.
4. Consider first the generating matrix G of the Reed-Solomon code. By the interpolation properties of polynomials, it is easy to see that any partition of the rows of G is a full-rank partition. In particular, the restriction of G to any $2^t/K_{out}$ blocks is a bijection mapping of the n bits to n bits, and hence the corresponding submatrix is invertible.

   Next, consider only the first n bits of the Reed-Solomon encoding corresponding to n/t blocks. (All other $(K_{out}-1)\times(n/t)$ blocks are handled similarly.)

   Note that applying C_{in} on each of the n/t blocks, corresponds to multiplying G by a block-diagonal matrix $M^* \in \mathbb{F}_2^{K_{in}\cdot t\cdot 2^t\times t\cdot 2^t}$, where each block of M^* is the matrix $M \in \mathbb{F}_2^{K_{in}t\times t}$.

Consider the partition $(\{1,\dots,t\},\{t+1,\dots,2t\},\dots\{(K_{in}-1)t+1,\dots K_{in}t\})$ of M, and denote $P_i=\{(i-1)t+1,(i-1)t+2,\dots,it\}$.
We claim that for each $1\le i\le K_{in}$, if we take the union of the rows of M^* corresponding to P_i's in each of the n/t blocks, then we get a full rank submatrix. That is because the n row are divided into n/t blocks on the diagonal of M^*, such that each of consists of a $t\times t$ full-rank submatrix (one for each P_i), and the rows are linearly independent of each other. Multiplying the first n rows of G (which is a submatrix of full-rank) by that submatrix of M^* (which is also full-rank) gives a full-rank submatrix. This defines a full-rank partition of C.

This completes the proof of Lemma 19. ◀

Proof of Theorem 15. Consider the code C_0 and the matrix M_0 from Claim 18 with $t=\Theta(\log\log(n))$ and $\delta=\epsilon/4$. By the guarantee of Claim 18, M_0 satisfies the desired properties with probability at least $2^{-O(1/\epsilon^2)}$. Assuming that we have such M_0, the encoding algorithm of C_0 runs in time $E_0(t)=O(t^2/\epsilon^2)$, and the trivial list-decoding works in time $D_0(t)=2^t\cdot\mathrm{poly}(t/\epsilon^2)$ by simply encoding all possible messages and checking their distance from the given word.

We construct C in two steps.

1. First, we apply Lemma 19 with C_0 as the inner code, obtaining the code

$$C_1\colon \mathbb{F}_2^{t2^t/\mathrm{poly}(1/\epsilon)}\to\mathbb{F}_2^{t2^t\cdot\mathrm{poly}(1/\epsilon)}$$

such that
   - **(i)** C_1 is $(1/2-\epsilon/2,\mathrm{poly}(1/\epsilon))$-list decodable,
   - **(ii)** the running time of the encoding algorithm is $E_1=\tilde{O}(2^t\cdot E_0(t))=\tilde{O}(2^t)\cdot\mathrm{poly}(1/\epsilon)$,
   - **(iii)** the decoding algorithm runs in time $D_1=D_0(t)\cdot 2^t+\mathrm{poly}(1/\epsilon)\cdot\widetilde{O}(t\cdot 2^t)=\tilde{O}(2^t)\cdot\mathrm{poly}(1/\epsilon)$, and
   - **(iv)** C_1 has a full-rank partition.
2. Next we apply Lemma 19 with C_1 as the inner code. We obtain the code $C\colon\mathbb{F}_2^n\to\mathbb{F}_2^{\mathrm{poly}(1/\epsilon)n}$, where $n=t\cdot 2^t/\mathrm{poly}(1/\epsilon)\cdot 2^{t\cdot 2^t/\mathrm{poly}(1/\epsilon)}/\mathrm{poly}(1/\epsilon)=2^{2^t/\mathrm{poly}(1/\epsilon)}$. By the guarantees of Lemma 19, C satisfies the following properties:
   - **(i)** it is $(1/2-\epsilon,\mathrm{poly}(1/\epsilon))$-list decodable,
   - **(ii)** the running time of the encoding algorithm is $\widetilde{O}(E_1\cdot n)=\mathrm{poly}(1/\epsilon)\cdot\widetilde{O}(n)$,
   - **(iii)** the running time of the list decoding algorithm is upper bounded by $D_1\cdot\mathrm{poly}(1/\epsilon)\cdot\widetilde{O}(n)=\mathrm{poly}(1/\epsilon)\cdot\widetilde{O}(n)$, and
   - **(iv)** C has a full-rank partition.

This completes the proof of Theorem 15. ◀

▶ Remark 20. It is not difficult to generalize Theorem 15 to a finite field $\mathbb{F}_p$ for any prime p. To extend Claim 18 to a finite field of order p, apply the result of Guruswami, Håstad, Kopparty [14, Theorem 2.5] that the random linear code $C_0\colon\mathbb{F}_p^t\to\mathbb{F}_p^{Kt}$ is $(1-1/p-\delta,\mathrm{poly}(1/\delta))$-list decodable with probability $1-2^{-\mathrm{poly}(\delta)t}$. Its generating matrix M will have a full-rank partition with probability greater than 0.288^K. The rest of the construction goes through in a similar manner.