

Price of Precision in Coded Distributed Matrix Multiplication: A Dimensional Analysis

Junge Wang, Zhuqing Jia and Syed A. Jafar
 Center for Pervasive Communications and Computing (CPCC)
 University of California, Irvine
 Email: {jungew, zhuqingj, syed}@uci.edu

arXiv:2105.07567v1 [cs.IT] 17 May 2021

Abstract—Coded distributed matrix multiplication (CDMM) schemes, such as MatDot codes, seek efficient ways to distribute matrix multiplication task(s) to a set of N distributed servers so that the answers returned from any R servers are sufficient to recover the desired product(s). For example, to compute the product of matrices \mathbf{U}, \mathbf{V} , MatDot codes partition each matrix into $p > 1$ sub-matrices to create smaller coded computation tasks that reduce the upload/storage at each server by $1/p$, such that \mathbf{UV} can be recovered from the answers returned by any $R = 2p - 1$ servers. An important concern in CDMM is to reduce the recovery threshold R for a given storage/upload constraint. Recently, Jeong et al. introduced Approximate MatDot (AMD) codes that are shown to improve the recovery threshold by a factor of nearly 2, from $2p - 1$ to p . A key observation that motivates our work is that the storage/upload required for approximate computing depends not only on the dimensions of the (coded) sub-matrices that are assigned to each server, but also on their precision levels — a critical aspect that is not explored by Jeong et al. Our main contribution is a rudimentary dimensional analysis of AMD codes inspired by the Generalized Degrees of Freedom (GDoF) framework previously developed for wireless networks, which indicates that for the same upload/storage, once the precision levels of the task assignments are accounted for, AMD codes surprisingly fall short in all aspects of even the trivial replication scheme which assigns the full computation task to every server. Indeed, the trivial replication scheme has a much better recovery threshold of 1, better download cost, better computation cost, and much better encoding/decoding (none required) complexity than AMD codes. The dimensional analysis is supported by simple numerical experiments.

I. INTRODUCTION

Coded distributed matrix multiplication (CDMM) [1]–[26] (see Figure 1) seeks to distribute a matrix multiplication task among N servers as efficiently as possible so that from the answers received from any R responsive servers the sink (user) is able to recover the desired computation result. R is referred to as *recovery threshold*. Existing state-of-the-art solutions [1], [2], [4] to CDMM are built upon matrix partitioning and polynomial based coding – the constituent matrices \mathbf{U}, \mathbf{V} are partitioned into block submatrices, coded shares of which are sent to the servers. The servers compute the products of their encoded shares, which can be viewed as evaluations of carefully constructed polynomials with partitioned block matrices as coefficients. Categorized by different partitioning strategies, state-of-the-art approaches fall into three classes: Polynomial codes [1] for row-by-column partitioning, MatDot codes [2] for column-by-row partitioning and Entangled Polynomial codes (EP codes) [4] for arbitrary partitioning.

When CDMM schemes are utilized for computations over real numbers, numerical stability concerns become important [27]–[29]. As noted in the literature [30], [31], this is because real Vandermonde matrices (which are essential in decoding) are ill-conditioned, especially for large R . In other words, small error in answers (that is a natural result of quantization) yields large error in decoded computation results. To overcome this problem, a variety of techniques are developed, for example, Chebyshev polynomial based coding schemes [27], circulant and rotation matrix embeddings [28], and random Khatri-Rao product codes [29].

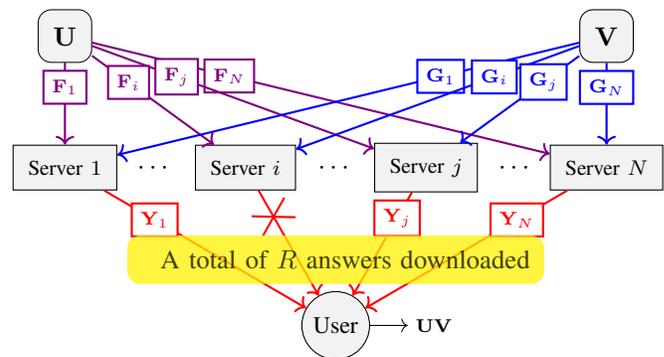


Fig. 1: Coded Distributed Matrix Multiplication (CDMM).

Another important concern in CDMM literature has been to find ways to reduce the recovery threshold for a given storage/upload cost per server. In this regard, a recent breakthrough is reported in [32] which introduces approximated CDMM solutions under the polynomial based coding framework. Based on MatDot codes¹ and a set of sufficiently small evaluation points, [32] shows that Approximate MatDot (AMD) codes achieve the recovery threshold of $R = p$ with bounded error ϵ . Compared with MatDot codes where the achieved recovery threshold is $R = 2p - 1$, this is “nearly twice as efficient as exact multiplication”.

Since a repetition code scheme which replicates the full computation task at every server can trivially achieve the recovery threshold of $R = 1$, it is crucial that the recovery threshold be optimized subject to constraints on the upload or storage cost. On the one hand, because AMD codes use the same matrix partitioning by a factor of $1/p$ as MatDot codes,

¹As in [32], the results generalize to ϵ -approximate EP codes as well.

one might imagine that just like MatDot codes, AMD codes require a fraction $1/p$ of the storage/upload cost (compared to repetition codes). But on the other hand, when approximate computation is involved, the storage/upload cost also depends strongly on the precision with which numerical values need to be represented. Our work is motivated by the goal of properly accounting for the scaling of upload/storage costs as a function of the desired precision of computation.

To accomplish this goal we take an approach inspired by the Generalized Degrees of Freedom (GDoF) framework that has been extensively used [33], [34] to study the approximate and robust capacity limits of wireless networks. Based on a similar framework, our rudimentary dimensional analysis reveals insights consistent with [32] in terms of the required ‘small’ size of evaluation points.² Surprisingly, it also reveals a strongly pessimistic outlook of AMD codes. The advantage of reduced recovery threshold that is achieved by AMD codes, is shown to come at the cost of increased upload/storage costs by a factor of p , due to increased precision requirement from the uploads. Intuitively, this is because the small evaluation points of AMD codes produce extremely ill-conditioned decoding Vandermonde matrices. In fact the dimensional analysis shows that AMD codes give away so much in their upload/storage costs that they fall short of even replication codes in all regards. Indeed, the trivial replication scheme has a much better recovery threshold of 1, better download cost, better computation cost, and much better encoding/decoding (none required) complexity than AMD codes. The dimensional analysis is supported by modest numerical experiments.

Notation: For integers m, n such that $m < n$, $[m : n] \triangleq \{m, m+1, \dots, n\}$, $\mathbf{X}_{m:n} \triangleq \{\mathbf{X}_m, \mathbf{X}_{m+1}, \dots, \mathbf{X}_n\}$. $[n] \triangleq [1 : n]$. $\|\cdot\|_F$ denotes the Frobenius norm. The notation $\tilde{O}(a \log b)$ suppresses³ polylog terms. When an $m \times p$ matrix \mathbf{A} is multiplied by a $p\lambda \times n\kappa$ matrix \mathbf{B} where \mathbf{B} is written as a block matrix with $p \times n$ blocks and the blocks have the dimension $\lambda \times \kappa$, the product $\mathbf{A}\mathbf{B}$ means that $(\mathbf{A} \otimes \mathbf{I}_\lambda)\mathbf{B}$ where \otimes is the Kronecker product and \mathbf{I}_λ is the $\lambda \times \lambda$ identity matrix.

II. PRELIMINARIES

A. MatDot Codes [2]

Partition matrices \mathbf{U}, \mathbf{V} as follows,

$$\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_p], \mathbf{V} = [\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_p]^T \quad (1)$$

Let $\alpha_1, \alpha_2, \dots, \alpha_N$ be distinct elements in \mathbb{R} . Server i , upon receiving the encoded matrices,

$$\mathbf{F}_i = \mathbf{U}_1 + \alpha_i \mathbf{U}_2 + \dots + \alpha_i^{p-1} \mathbf{U}_p \quad (2)$$

$$\mathbf{G}_i = \mathbf{V}_p + \alpha_i \mathbf{V}_{p-1} + \dots + \alpha_i^{p-1} \mathbf{V}_1, \quad (3)$$

²Our analysis can also be extended to ‘large’ evaluation points, but since the conclusion remains pessimistic, that case is omitted here.

³There is another standard definition of the notation \tilde{O} which fully suppresses polylog terms, i.e. $O(\text{apolylog}(b))$ is represented by $\tilde{O}(a)$. The definition used in this paper emphasizes the dominant factor in the polylog term.

computes the product,

$$\mathbf{C}_i = \mathbf{F}_i \mathbf{G}_i = \sum_{i=1}^p \sum_{j=1}^p \mathbf{U}_i \mathbf{V}_j \alpha_i^{p-1+i-j}. \quad (4)$$

The product can be viewed as a polynomial (with respect to α) of degree $2p-2$. With the answers from any $R=2p-1$ servers, the user is able to recover all the coefficients of the polynomial, so that the desired matrix product $\mathbf{C} = \mathbf{U}\mathbf{V} = \sum_{i=1}^p \mathbf{U}_i \mathbf{V}_i$ can be recovered since it is the coefficient of the term α^{p-1} . The recovery threshold is $R(p, 0) = 2p-1$. The notation is explained in the next section.

B. Approximate Computation [32]

Assume $\|\mathbf{U}\|_F \leq \eta, \|\mathbf{V}\|_F \leq \eta$. The goal is to perform approximate computation so that

$$|\hat{\mathbf{C}}_{i,j} - \mathbf{C}_{i,j}| \leq \epsilon \quad (5)$$

The recovery threshold subject to the constraint (5), is denoted as $R(p, \epsilon)$. [32] introduces Approximate MatDot codes that are able to achieve the ‘optimal’ recovery threshold $R(p, \epsilon) = p$. The main result of [32] is summarized below.

Theorem 1. [32] (*Achievability*) For any $0 \leq \epsilon \leq \min(2, 3\eta^2\sqrt{2p-1})$, AMD codes can be constructed by choosing the evaluation points $\alpha_1, \alpha_2, \dots, \alpha_N$ as

$$|\alpha_i| \leq \frac{\epsilon}{6\eta^2\sqrt{2p-1}(p^2-p)} \quad (6)$$

such that $R(p, \epsilon) = p$.

(Converse) For all $0 \leq \epsilon \leq \eta^2$, $R(p, \epsilon) \geq p$

Remark 1. Intuitively, the key idea of AMD codes is to assign small values to the evaluation points α_i , so that the terms with sufficiently high powers of α_i become essentially negligible, thus reducing the number of unknowns, which in turn reduces the recovery threshold.

III. THE PRICE OF PRECISION IN CDMM

A. GDoF Framework

Inspired by the GDoF framework that has been used extensively [33], [34] to study the approximate capacity of wireless networks let us introduce a similar basic framework to study the fundamental tradeoffs in the approximate computing problem. As a toy example to introduce basic notation, using base- B representation⁴ for numerical values, consider a random variable that takes values, say in $[0, B^\mu)$, $B=10, \mu=4$, and let W be its representation accurate to $\nu=5$ digit precision. W may be represented as (cf. [34]),

$$W = B^\mu (\overline{W} + B^{-\nu} \tilde{W}) \quad (7)$$

with $\overline{W}, \tilde{W} \in [0, 1)$, such that $B^\mu \overline{W}$ represents the actual value (with infinite precision), $B^{-\nu} \tilde{W}$ represents noise that primarily affects (truncates or subtracts) the

⁴GDoF studies [34] of wireless networks use base P representation, and P is labeled ‘power’ as a legacy from prior DoF studies where it indeed represents transmit power.

digits that appear after the ν digits that are accurately known, and W represents the truncated value. For example, $\mathbf{w}_1\mathbf{w}_2\mathbf{w}_3\mathbf{w}_4.\mathbf{w}_5 = 10^4(0.\mathbf{w}_1\mathbf{w}_2\mathbf{w}_3\mathbf{w}_4.\mathbf{w}_5w_6w_7\cdots + 10^{-5}(-0.w_6w_7\cdots))$, is a 5 digit precision representation of the actual value $\mathbf{w}_1\mathbf{w}_2\mathbf{w}_3\mathbf{w}_4.\mathbf{w}_5w_6w_7\cdots$ where the precise digits are highlighted in bold. Note that the noise term $B^{-\nu}\tilde{W}$ is not independent of \overline{W} . Equivalently, one may view W as the truncated version of \overline{W} , carrying only the ν digits of \overline{W} that are accurately known, i.e.,

$$W = B^{\mu-\nu}(\overline{W})^\nu, \quad (8)$$

where we define the notation,

$$(\overline{W})^\nu \triangleq \lfloor B^\nu \overline{W} \rfloor, \quad (9)$$

to represent the integer value comprised of the top ν digits of the normalized quantity \overline{W} . This will be the standard notation throughout this work.

The GDoF framework uses such exponential representations in a generic B -ary alphabet, and allows the base B to approach infinity. This has the advantage that it removes lower order effects (because of normalization by $\log_2(B)$ when measuring information in B -ary units), thus smoothing out the finer details, e.g., the choice of the particular input distribution, which do not scale with alphabet size, and exposes the sharp fundamental tradeoffs that are typically expected from dimensional analysis. In fact, for the GDoF framework, instead of the interval $[0, 1)$ it suffices to assume that \overline{W} and \tilde{W} are $O(1)$, i.e., bounded by some constants independent of B . We will use the overline and tilde notations throughout this work to represent the normalized (limited to $O(1)$) precise values and noise, and μ_\bullet, ν_\bullet for the magnitude and precision levels, respectively.

B. Dimensional Analysis: $\min(\nu_f, \nu_g) \geq p\nu$

As the main result of this work we will show in this section that for AMD codes the uploads to each server $(\mathbf{F}_i, \mathbf{G}_i)$ need precision levels (ν_f, ν_g) that are at least p times greater than the precision level of the computed product (ν) . In other words, the main result is the bound,

$$\min(\nu_f, \nu_g) \geq p\nu.$$

Using notation consistent with the GDoF formulation, the uploads to the servers are represented as

$$\mathbf{F}_i = B^{\mu_f}(\overline{\mathbf{F}}_i + B^{-\nu_f}\tilde{\mathbf{F}}_i) = B^{\mu_f-\nu_f}(\overline{\mathbf{F}}_i)^{\nu_f} \quad (10)$$

$$\mathbf{G}_i = B^{\mu_g}(\overline{\mathbf{G}}_i + B^{-\nu_g}\tilde{\mathbf{G}}_i) = B^{\mu_g-\nu_g}(\overline{\mathbf{G}}_i)^{\nu_g} \quad (11)$$

where

$$\overline{\mathbf{F}}_i = \overline{\mathbf{U}}_1 + \alpha_i\overline{\mathbf{U}}_2 + \cdots + \alpha_i^{p-1}\overline{\mathbf{U}}_p \quad (12)$$

$$\overline{\mathbf{G}}_i = \alpha_i^{p-1}\overline{\mathbf{V}}_1 + \alpha_i^{p-2}\overline{\mathbf{V}}_2 + \cdots + \overline{\mathbf{V}}_p. \quad (13)$$

It is assumed that the normalized inputs $\overline{\mathbf{U}}_i, \overline{\mathbf{V}}_i$, and the noise terms $\tilde{\mathbf{F}}_i, \tilde{\mathbf{G}}_i$ are all $O(1)$. Thus, the uploads $\mathbf{F}_i, \mathbf{G}_i$ have precision ν_f, ν_g , respectively. The scaling factors B^{μ_f}, B^{μ_g} can be normalized away in this setting, but let us keep them for

an explicit representation of the scale of \mathbf{F}_i and \mathbf{G}_i . Following the insights of [32], the distinct constants α_i are assumed to be small, say

$$\alpha_i = B^{-\delta\overline{\alpha}_i} \quad (14)$$

for some $\delta > 0$ and $\overline{\alpha}_i = \Theta(1)$. The products $\mathbf{F}_i\mathbf{G}_i$ computed by the servers are expressed as follows.

$$\mathbf{F}_i\mathbf{G}_i = B^{\mu_f+\mu_g} \left(\overline{\mathbf{F}}_i\overline{\mathbf{G}}_i + B^{-\nu_f}\tilde{\mathbf{F}}_i\overline{\mathbf{G}}_i + B^{-\nu_g}\overline{\mathbf{F}}_i\tilde{\mathbf{G}}_i + B^{-\nu_f-\nu_g}\tilde{\mathbf{F}}_i\tilde{\mathbf{G}}_i \right) \quad (15)$$

which has precision limited to $\min(\nu_f, \nu_g)$ digits because of the additional noise terms. From Server i , the user downloads $\mathbf{F}_i\mathbf{G}_i$ to ν_y digit precision. Since $\mathbf{F}_i\mathbf{G}_i$ has only $\min(\nu_f, \nu_g)$ digit precision, we require

$$\nu_y \leq \min(\nu_f, \nu_g). \quad (16)$$

The download from Server i is then represented as

$$\mathbf{Y}_i = B^{\mu_f+\mu_g}(\overline{\mathbf{Y}}_i + B^{-\nu_y}\tilde{\mathbf{Y}}_i) = B^{\mu_f+\mu_g-\nu_y}(\overline{\mathbf{Y}}_i)^{\nu_y} \quad (17)$$

$$\overline{\mathbf{Y}}_i = \overline{\mathbf{F}}_i\overline{\mathbf{G}}_i \quad (18)$$

$$= \overline{\mathbf{U}}_1\overline{\mathbf{V}}_p + \alpha_i(\overline{\mathbf{U}}_1\overline{\mathbf{V}}_{p-1} + \overline{\mathbf{U}}_{p-1}\overline{\mathbf{V}}_p) + \cdots + \alpha_i^{p-1}(\overline{\mathbf{U}}_1\overline{\mathbf{V}}_1 + \cdots + \overline{\mathbf{U}}_p\overline{\mathbf{V}}_p) + \alpha_i^{2p-2}(\overline{\mathbf{U}}_p\overline{\mathbf{V}}_1) \quad (19)$$

$$= \overline{\mathbf{X}}_0 + \alpha_i\overline{\mathbf{X}}_1 + \cdots + \alpha_i^{2p-2}\overline{\mathbf{X}}_{2p-2} \quad (20)$$

The compact notation $\overline{\mathbf{X}}_i$ is used for the $O(1)$ terms that represent the corresponding sums of various $\overline{\mathbf{U}}_i\overline{\mathbf{V}}_j$ terms. The desired term is $\overline{\mathbf{X}}_{p-1}$.

A recovery threshold of p means that decoding must be accomplished with only p server responses. Without loss of generality, say we have the responses $\mathbf{Y}_1, \dots, \mathbf{Y}_p$.

$$\begin{aligned} \begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \vdots \\ \mathbf{Y}_p \end{bmatrix} &= B^{\mu_f+\mu_g} \begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{p-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{p-1} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \alpha_p & \cdots & \alpha_p^{p-1} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{X}}_0 \\ \overline{\mathbf{X}}_1 \\ \vdots \\ \overline{\mathbf{X}}_{p-1} \end{bmatrix} \\ &+ B^{\mu_f+\mu_g} \begin{bmatrix} \alpha_1^p & \cdots & \alpha_1^{2p-2} \\ \alpha_2^p & \cdots & \alpha_2^{2p-2} \\ \vdots & \cdots & \vdots \\ \alpha_p^p & \cdots & \alpha_p^{2p-2} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{X}}_p \\ \overline{\mathbf{X}}_{p+1} \\ \vdots \\ \overline{\mathbf{X}}_{2p-2} \end{bmatrix} + B^{\mu_f+\mu_g-\nu_y} \begin{bmatrix} \tilde{\mathbf{Y}}_1 \\ \tilde{\mathbf{Y}}_2 \\ \vdots \\ \tilde{\mathbf{Y}}_p \end{bmatrix} \end{aligned} \quad (21)$$

From here, with certain ('mild') additional assumptions (see Section VI and Section VII) it can be information theoretically argued that the desired quantity $\overline{\mathbf{X}}_{p-1} = \overline{\mathbf{U}}\overline{\mathbf{V}}$ can be recovered from \mathbf{Y} with precision no higher than $(\min(\nu_y, p\delta) - (p-1)\delta)^+$. We defer the information theoretic derivation to Section VII at the end of this paper, and provide here an intuitive justification instead, as follows.

Solving for $\overline{\mathbf{X}}_{p-1} = \overline{\mathbf{U}}\overline{\mathbf{V}}$ involves an inversion of the first Vandermonde matrix while treating the other terms as noise. The first Vandermonde matrix has condition number at least $\Omega(B^{(p-1)\delta})$, which causes a noise amplification by that factor in the remaining terms. Intuitively, since $\overline{\mathbf{X}}_0, \overline{\mathbf{X}}_1, \dots, \overline{\mathbf{X}}_{p-2}$

dominate $\bar{\mathbf{X}}_{p-1}$, this amounts to projection of the \mathbf{Y} vector along the vector that lies in the null space of the first $p-1$ columns of the first Vandermonde matrix. In this projected dimension, \mathbf{X}_{p-1} can be recovered as the dominant term, and the remaining noise level is determined by the stronger of the two projected noise terms: the projection of $\tilde{\mathbf{Y}}$, which has strength $B^{\mu_f + \mu_g + (p-1)\delta - \nu_y}$, and the projection of $\bar{\mathbf{X}}_p$, which has strength $B^{\mu_f + \mu_g + (p-1)\delta - p\delta}$. Note that the $B^{(p-1)\delta}$ scaling factor appears in each case due to the noise amplification impact of the inversion of the first Vandermonde matrix. This allows the user to recover

$$\mathbf{UV} = \overline{\mathbf{UV}} + B^{(p-1)\delta - \min(\nu_y, p\delta)} \widetilde{\mathbf{UV}} \quad (22)$$

Thus, the answer can be recovered with ν digit precision, provided that,

$$\nu \leq \min(\nu_y, p\delta) - (p-1)\delta \quad (23)$$

$$\leq \min(\nu_f, \nu_g, p\delta) - (p-1)\delta \quad (24)$$

$$\implies \begin{cases} \nu \leq \nu_f - (p-1)\delta \\ \nu \leq \nu_g - (p-1)\delta \\ \nu \leq \delta \end{cases} \quad (25)$$

Thus, the dimensional analysis yields a bound on the required precision of the uploads as,

$$\nu_f \geq \nu + (p-1)\delta \quad (26)$$

$$\geq \nu + (p-1)\nu \quad (27)$$

$$= p\nu \quad (28)$$

Similarly, $\nu_g \geq p\nu$.

IV. OBSERVATIONS

Let us interpret the result of [32] that is summarized as Theorem 1 in this paper, in GDoF terms. To this end, let $\eta = B^{\mu/2}$, $\epsilon = B^{\mu-\nu}$, where ν can be regarded as the precision level. Thus the entries of \mathbf{U}, \mathbf{V} are of the order $B^{\mu/2}$, so that the entries of \mathbf{C} are of the order B^μ . The precision level of each entry is at least ν digits in the B -ary alphabet. [32] shows that to evaluate the matrix product to ν digit precision, the choice of the α_i should satisfy condition (6) which is restated as follows in GDoF terms,

$$|\alpha_i| = B^{-\delta} \bar{\alpha}_i \leq \frac{B^{\mu-\nu}}{6B^\mu \sqrt{2p-1}(p^2-p)} = O(B^{-\nu}) \quad (29)$$

$$\implies \delta \geq \nu \quad (30)$$

This is indeed one of the conditions that we find from our dimensional analysis as well, as it appears in (25).

The dimensional analysis goes a bit further, and reveals a rather pessimistic outlook according to which AMD codes fall short of even trivial repetition codes in all aspects. A repetition code refers to the scheme that assigns the full computation task to each server by uploading the entire \mathbf{U}, \mathbf{V} matrices to each server, with ν digit precision, and downloads the result of the computation from any 1 server, also to ν digit precision. On the other hand, AMD codes upload submatrices that are smaller by $1/p$ in terms of their number of elements, but with precision $p\nu$ for each element, which is p times larger, so they have

the same upload/storage cost as repetition codes. Moreover, in terms of recovery threshold, computation cost, and download cost, AMD codes are strictly worse than repetition codes.

Repetition codes have a recovery threshold of 1 because the download from any 1 server suffices. While repetition codes download the answer \mathbf{UV} as a $\lambda \times \lambda$ matrix to ν digit precision from only one server, AMD codes download a $\lambda \times \lambda$ matrix from each of p servers, in each case to $p\nu$ precision, so the download cost of AMD codes is p^2 times greater than repetition codes. In terms of computation cost, recall that the complexity of multiplying two n digit numbers is super-linear in n — the trivial multiplication scheme has complexity $O(n^2)$ but the Schönhage—Strassen algorithm reduces it to $\tilde{O}(n \log n)$ which is still superlinear. Now, AMD codes require fewer multiplications by a factor of $1/p$ due to matrix partitioning, however, since each multiplication is between numbers with a greater number of digits by a factor of p , and the complexity of multiplication is super linear in the number of digits, it turns out that AMD codes require greater computation complexity at each server, compared to repetition codes. Specifically, AMD codes require p times fewer multiplications than repetition codes but each multiplication has complexity $\tilde{O}(p\nu \log(p\nu))$ for AMD codes, as compared to $\tilde{O}(\nu \log \nu)$ for repetition codes. The comparison is illustrated in Table I. While encoding and decoding complexities are not listed in the table, note that repetition codes do not require encoding/decoding at all, so AMD codes fall short of repetition codes in this regard as well.

	MatDot codes	AMD codes	Repetition codes
Recovery threshold	$2p-1$	p	1
Upload/storage per server	ν/p	ν	ν
Total Download	$(2p-1)\nu$	$p^2\nu$	ν
Computation cost per server	$\tilde{O}(\frac{\nu \log \nu}{p})$	$\tilde{O}(\nu \log(p\nu))$	$\tilde{O}(\nu \log \nu)$

TABLE I: *MatDot codes vs AMD codes vs repetition codes. Values shown are relative to each other.*

Last but not the least, it is important to also note the caveat that while dimensional analysis allows elegant characterizations of fundamental tradeoffs, this elegance relies on asymptotic analysis that neglects lower order effects. As such, in settings where the lower order effects are important, e.g., where matrices comprised of small numbers are being multiplied to low precision so the large B assumption is not justified, it is conceivable that the conclusions of the dimensional analysis may be violated. While dimensional analysis informs our intuition and provides principled reasoning at a high level, ultimately numerical results are still important to fully reveal the finer tradeoffs for particular settings. Elaborate experiments are beyond the scope of this work, but modest numerical results are provided next, that indeed validate the insights from the dimensional analysis.

V. NUMERICAL RESULTS

Consider a simple setting where $p = 3$, and the dimensions of the \mathbf{U} , \mathbf{V} matrices are 1×3 and 3×1 , respectively, $\mathbf{U} = [U_1, U_2, U_3]$, $\mathbf{V} = [V_1, V_2, V_3]^T$, where $U_1, U_2, U_3, V_1, V_2, V_3$ are uniformly i.i.d. over $[0, 1]$. We use base $B = 10$, i.e., decimal representations. The recovery threshold of AMD codes for this setting is $R = 3$, and to simplify⁵ our simulation, we consider the setting $N = R = 3$. The encoded version of the constituent matrices for Server i are the following two scalars.

$$F_i = \text{truncate}(U_1 + \alpha_i U_2 + \alpha_i^2 U_3, \gamma), \quad (31)$$

$$G_i = \text{truncate}(V_3 + \alpha_i V_2 + \alpha_i^2 V_1, \gamma), \quad (32)$$

where the function $\text{truncate}(x, \gamma)$ truncates the value of x at γ digits after the decimal. The answer returned by Server i is $Y_i = \text{truncate}(F_i G_i, \gamma)$. Denote $\max_{i \in [N]} \alpha_i$ as α_{\max} , the selection of evaluation nodes $\alpha_1, \alpha_2, \alpha_3$ is given as $\alpha_i = \frac{i}{N} \alpha_{\max}, \forall i \in [N]$. We use the decoding algorithm⁶ [32, (55)], i.e., the minimum norm solution to decode ϵ -approximate MatDot codes.

Figure 2 plots the Monte Carlo simulation results of upload/download cost per server (i.e., γ) versus mean absolute error (MAE) for $\gamma \in \{4, 5, \dots, 16\}$ and $\alpha_{\max} = 10^{-4}$. It is evident that to achieve the desired approximation error of 10^{-4} , i.e., $\nu = 4$, the upload/download cost per server required is at least $\gamma = 12 = 3 \times 4 = p\nu$, which confirms our analytical result. Figure 3 plots Monte Carlo simulation results of α_{\max} versus MAE for $\alpha_{\max} \in \{10^{-7}, 10^{-6}, \dots, 10^{-1}\}$ and $\gamma = 12$. Simulation results show that for the given upload/download cost per server $\gamma = 12$, the best approximation error is achieved when $\alpha_{\max} = 10^{-4} = 10^{-12/3} = 10^{-\gamma/p}$. Since given $\gamma = 12$, as illustrated in the red line in Figure 3, repetition codes (which do not depend on the selection of α_{\max}) achieve the MAE of no more than 10^{-4} with the same upload cost, this again confirms our analytical results that ϵ -approximate MatDot codes fall short of repetition codes.

VI. CONCLUSION

The nature of our analysis is that of a converse argument, i.e., an impossibility result, which is only as strong as the generality with which it applies. So it is important to note its limitations. For instance, the information theoretic analysis in Section VII assumes $\bar{\mathbf{X}}_i$ are independent and scalars, but neither of those assumptions is beyond reproach. Indeed, while $\bar{\mathbf{X}}_i$ are perhaps algebraically independent, they may not be statistically independent, and in general they can certainly be matrices. The assumptions of Section VII are still meaningful, in that the converse applies to any scheme that does not take advantage of any potential dependence between $\bar{\mathbf{X}}_i$ and that

⁵Note that indeed, this is the best-case scenario for AMD codes. When there are stragglers, i.e., $N > R$, the condition number of the corresponding decoding matrix is even worse.

⁶To completely characterize the trade-off between upload/download costs, approximation error ϵ and the choice of α_{\max} , we do not declare failure in our decoding algorithm even if the norm of the minimum norm solution exceeds the threshold $\sqrt{2p-1}\eta^2$ in Algorithm 1 of [32].

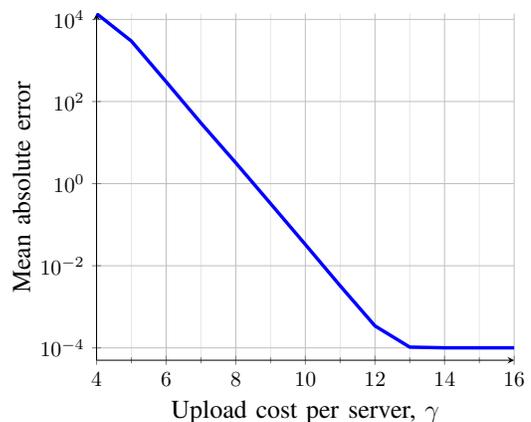


Fig. 2: Upload cost per server γ vs mean absolute error.

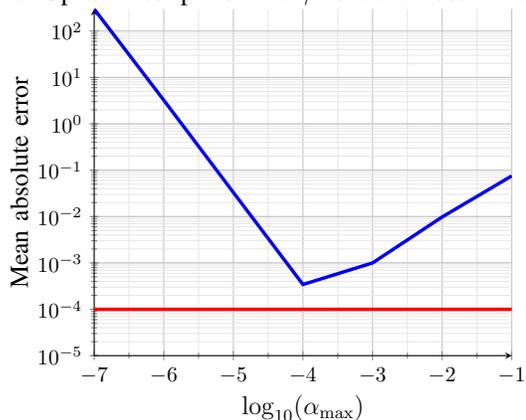


Fig. 3: $\log_{10}(\alpha_{\max})$ versus mean absolute error.

decodes each element of the $\bar{\mathbf{X}}_{p-1}$ matrix by the same decoding rule. To our knowledge, this is true for all existing CDMM schemes, including AMD codes, and is likely to be true for most future schemes as well, because the applications for CDMM typically require low decoding complexity. Similarly, aside from their magnitude constraints, the evaluation points α_i are assumed ‘generic’, which is also true for all known schemes, but specialized choices of α_i that achieve alignment may be possible. This possibility is reminiscent of rational alignment in wireless networks [35]. On the other hand, even if such constructions are possible, the limited precision aspect may negate their benefits, if analogies may be drawn from wireless GDoF studies [36]. Nevertheless, all such limitations of current analysis leave the door open for future surprises. So while the benefits of AMD codes are indeed called in question by current analysis, we do not expect this pessimistic outlook to be the final word along this new research avenue. On the contrary, we are optimistic that the idea of exploiting the power dimension, that is introduced by AMD codes, may find clever uses in coded computing to enable new forms of interference alignment [37], [38], just as the power dimension plays a critical role in the GDoF characterizations of wireless networks. Better formalizations of the GDoF perspective for distributed computing, that improve upon our rudimentary

attempt in this work, may be the key to future advances along this promising research avenue.

VII. APPENDIX

Let us make the simplifying assumption that $\bar{\mathbf{X}}_i, i \in [0 : 2p - 2]$ are independent scalars, and that there exists a finite constant Δ such that the joint differential entropy of any non-empty subset $S \subset \{\bar{\mathbf{X}}_i : i \in [0 : 2p - 2]\}$ is bounded as $h(S) > \Delta$ (bits). The assumption of independence of $\bar{\mathbf{X}}_i$ and that they are scalars may appear rather restrictive, because in practice $\bar{\mathbf{X}}_i$ may be neither, but the assumption is general enough to encompass any decoding scheme that does not exploit potential dependencies across $\bar{\mathbf{X}}_i$, and which applies the same decoding rule to recover each element of the $\bar{\mathbf{X}}_{p-1} = \overline{\mathbf{UV}}$ matrix. See Section VI for additional discussion of such limitations.

The ν digit precision of the recovered computed value is represented by the following bound on the mean absolute error distortion,

$$\mathbb{E}|\overline{\mathbf{UV}} - \mathbf{UV}| = O(B^{-\nu}). \quad (33)$$

Therefore, we have

$$I(\overline{\mathbf{UV}}; \mathbf{UV}) = h(\overline{\mathbf{UV}}) - h(\overline{\mathbf{UV}} | \mathbf{UV}) \quad (34)$$

$$\geq \Delta - h(\overline{\mathbf{UV}} - \mathbf{UV} | \mathbf{UV}) \quad (35)$$

$$= \nu \log(B) + o(\log(B)) \quad (36)$$

For step (36) we used the fact that Laplace distributions are (differential) entropy maximizers subject to a mean absolute deviation⁷ constraint. Now, if any decoding rule applied to $\mathbf{Y}_{1:p}$ recovers \mathbf{UV} which represents $\bar{\mathbf{X}}_{p-1} = \overline{\mathbf{UV}}$ to ν -digit precision, then we have the Markov Chain $\bar{\mathbf{X}}_{p-1} = \overline{\mathbf{UV}} \leftrightarrow \mathbf{Y}_{1:p} \leftrightarrow \mathbf{UV}$. From the GDoF perspective we have,

$$\nu \leq \lim_{B \rightarrow \infty} \frac{I(\overline{\mathbf{UV}}; \mathbf{UV})}{\log(B)} \quad (37)$$

$$\leq \lim_{B \rightarrow \infty} \frac{I(\bar{\mathbf{X}}_{p-1}; \mathbf{Y}_{1:p})}{\log(B)} \quad (38)$$

For cleaner notation we will occasionally suppress $o(\log(B))$ terms that are inconsequential for GDoF according to (38).

$$I(\bar{\mathbf{X}}_{p-1}; \mathbf{Y}_{1:p}) \leq I(\bar{\mathbf{X}}_{p-1}; \bar{\mathbf{Y}}_{1:p}) \quad (39)$$

$$= h(\bar{\mathbf{Y}}_{1:p}) - h(\bar{\mathbf{Y}}_{1:p} | \bar{\mathbf{X}}_{p-1}) \quad (40)$$

$$h(\bar{\mathbf{Y}}_{1:p}) = h(\bar{\mathbf{Y}}_1) + h(\bar{\mathbf{Y}}_2 | \bar{\mathbf{Y}}_1) + \dots + h(\bar{\mathbf{Y}}_p | \bar{\mathbf{Y}}_{1:p-1})$$

$$\leq -(0 + 1 + \dots + (p-1))\delta \log(B) + o(\log(B)) \quad (41)$$

This is because conditioning on $\bar{\mathbf{Y}}_{1:i-1}$ allows (Gaussian) elimination of $\bar{\mathbf{X}}_{0:i-2}$ terms from $\bar{\mathbf{Y}}_i$, leaving the dominant term as $B^{-(i-1)\delta} \bar{\mathbf{X}}_{i-1}$ whose bounded support limits its

⁷The choice of mean absolute error vs mean squared error in (33) is inconsequential from a GDoF perspective. For example, if the precision constraint (33) is framed instead in terms of mean squared error, i.e., $\text{MSE} = O(B^{-2\nu})$, the same GDoF bound is still obtained by using the fact that Gaussians are entropy maximizers subject to a variance constraint.

entropy to $-(i-1)\delta \log(B)$ (uniform distribution maximizes entropy). Next we bound the other entropy term.

$$h(\bar{\mathbf{Y}}_{1:p} | \bar{\mathbf{X}}_{p-1}) \geq h(\bar{\mathbf{Y}}_{1:p} | \bar{\mathbf{X}}_{p-1}, \bar{\mathbf{X}}_{p+1:2p-2}) \quad (42)$$

$$= h\left(\mathbf{Q} \begin{bmatrix} \bar{\mathbf{X}}_0 \\ \vdots \\ \bar{\mathbf{X}}_{p-2} \\ \bar{\mathbf{X}}_p \end{bmatrix}\right) = h(\bar{\mathbf{X}}_{0:p-2}, \bar{\mathbf{X}}_p) + \log |\det(\mathbf{Q})| \quad (43)$$

$$\geq \Delta + \log |\det(\mathbf{Q})| = \log |\det(\mathbf{Q})| + o(\log(B)), \quad (44)$$

where

$$\mathbf{Q} = \begin{bmatrix} 1 & \dots & \bar{\alpha}_1^{p-2} B^{-(p-2)\delta} & \bar{\alpha}_1^p B^{-p\delta} \\ 1 & \dots & \bar{\alpha}_2^{p-2} B^{-(p-2)\delta} & \bar{\alpha}_2^p B^{-p\delta} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \dots & \bar{\alpha}_p^{p-2} B^{-(p-2)\delta} & \bar{\alpha}_p^p B^{-p\delta} \end{bmatrix}. \quad (45)$$

Note that (43) follows by the assumption of independence. The determinant of \mathbf{Q} can be approximated as

$$|\det(\mathbf{Q})| = O(B^{-(0+1+2+\dots+(p-2)+p)\delta}). \quad (46)$$

Substituting this approximation into (44), we have

$$h(\bar{\mathbf{Y}}_{1:p} | \bar{\mathbf{X}}_{p-1}) \geq -(1 + 2 + \dots + (p-2) + p)\delta \log(B). \quad (47)$$

Combining (38), (40), (41) and (47), we have our first desired bound,

$$\nu \leq \delta. \quad (48)$$

Our next bound is obtained as follows.

$$I(\bar{\mathbf{X}}_{p-1}; \mathbf{Y}_{1:p}) = I(\bar{\mathbf{X}}_{p-1}; (\bar{\mathbf{Y}}_{1:p})^{\nu_y}) \quad (49)$$

$$\leq I\left(\bar{\mathbf{X}}_{p-1}; \left\{(\bar{\mathbf{X}}_i)^{(\nu_y - i\delta)^+}\right\}_{i \in [0:p-1]}\right) \quad (50)$$

$$= I\left(\bar{\mathbf{X}}_{p-1}; (\bar{\mathbf{X}}_{p-1})^{(\nu_y - (p-1)\delta)^+}\right) \quad (51)$$

$$\leq H((\bar{\mathbf{X}}_{p-1})^{(\nu_y - (p-1)\delta)^+}) \quad (52)$$

$$\leq (\nu_y - (p-1)\delta)^+ \log(B) + o(\log(B)) \quad (53)$$

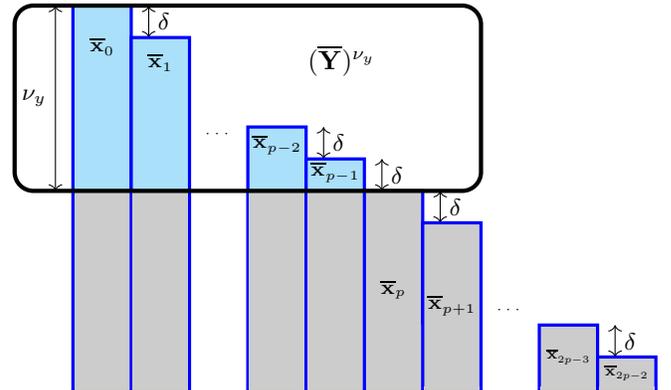


Fig. 4: The digit levels where various $\bar{\mathbf{X}}_i$ appear in \mathbf{Y}

Recall that $\mathbf{Y}_{1:p}$ is an invertible function of $(\bar{\mathbf{Y}}_{1:p})^{\nu_y}$. The key step (50) is explained by Figure 4 which shows that $(\bar{\mathbf{Y}}_{1:p})^{\nu_y}$ is in turn a function (up to bounded distortion which is inconsequential for GDoF) of the top ν_y digits of $\bar{\mathbf{X}}_0$, the top $(\nu_y - \delta)^+$ digits of $\bar{\mathbf{X}}_1$, the top $(\nu_y - 2\delta)^+$ digits of $\bar{\mathbf{X}}_2, \dots$, and the top $(\nu_y - (p-1)\delta)^+$ digits of $\bar{\mathbf{X}}_{p-1}$. Rigorous derivations of such bounds, while tedious, may be found in several recent works [39], so let us omit the details here. Combined with (38) this gives us our other desired bound:

$$\nu \leq (\nu_y - (p-1)\delta)^+. \quad (54)$$

REFERENCES

- [1] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4406–4416.
- [2] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2019.
- [3] S. Dutta, Z. Bai, H. Jeong, T. Low, and P. Grover, "A Unified Coded Deep Neural Network Training Strategy Based on Generalized PolyDot Codes for Matrix Multiplication," *ArXiv:1811.1075*, Nov. 2018.
- [4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.
- [5] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1215–1225.
- [6] A. Reiszadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.
- [7] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2418–2422.
- [8] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.
- [9] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Advances In Neural Information Processing Systems*, 2016, pp. 2100–2108.
- [10] —, "Coded convolution for parallel and distributed computing within a deadline," *arXiv preprint arXiv:1705.03875*, 2017.
- [11] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded fourier transform," *arXiv preprint arXiv:1710.06471*, 2017.
- [12] T. Jahani-Nezhad and M. A. Maddah-Ali, "Codedsketch: A coding scheme for distributed computation of approximated matrix multiplications," *arXiv preprint arXiv:1812.10460*, 2018.
- [13] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1993–1997.
- [14] G. Suh, K. Lee, and C. Suh, "Matrix sparsification for coded matrix multiplication," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 1271–1278.
- [15] S. Wang, J. Liu, N. Shroff, and P. Yang, "Fundamental limits of coded linear transform," *arXiv preprint arXiv:1804.09791*, 2018.
- [16] A. Mallick, M. Chaudhari, U. Sheth, G. Palanikumar, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–40, 2019.
- [17] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5152–5160.
- [18] A. Severinson, A. G. i Amat, and E. Rosnes, "Block-diagonal and It codes for distributed computing with straggling servers," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 1739–1753, 2018.
- [19] F. Haddadpour and V. R. Cadambe, "Codes for distributed finite alphabet matrix-vector multiplication," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1625–1629.
- [20] U. Sheth, S. Dutta, M. Chaudhari, H. Jeong, Y. Yang, J. Kohonen, T. Roos, and P. Grover, "An application of storage-optimal matdot codes for coded matrix multiplication: Fast k-nearest neighbors estimation," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1113–1120.
- [21] H. Jeong, F. Ye, and P. Grover, "Locally recoverable coded matrix multiplication," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2018, pp. 715–722.
- [22] M. Kim, J.-y. Sohn, and J. Moon, "Coded matrix multiplication on a group-based model," *arXiv preprint arXiv:1901.05162*, 2019.
- [23] H. Park, K. Lee, J.-y. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," *arXiv preprint arXiv:1801.04686*, 2018.
- [24] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 34–40, 2017.
- [25] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch computation," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2821–2846, 2021.
- [26] Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar, "Gcsa codes with noise alignment for secure coded multi-party batch matrix multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 306–316, 2021.
- [27] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2758–2785, 2021.
- [28] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," *arXiv preprint arXiv:1910.06515*, 2019.
- [29] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random khatri-rao-product codes for numerically-stable distributed matrix multiplication," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 253–259.
- [30] W. Gautschi and G. Inglese, "Lower bounds for the condition number of vandermonde matrices," *Numerische Mathematik*, vol. 52, no. 3, pp. 241–250, 1987.
- [31] V. Y. Pan, "How bad are vandermonde matrices?" *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 2, pp. 676–694, 2016.
- [32] H. Jeong, A. Devulapalli, V. R. Cadambe, and F. Calmon, "e-Approximate Coded Matrix Multiplication is Nearly Twice as Efficient as Exact Multiplication," *arXiv e-prints*, p. arXiv:2105.01973, May 2021.
- [33] R. Etkin, D. Tse, and H. Wang, "Gaussian interference channel capacity to within one bit," *IEEE Transactions on Information Theory*, vol. 54, no. 12, pp. 5534–5562, 2008.
- [34] A. Gholami Davoodi, B. Yuan, and S. A. Jafar, "GDoF region of the MISO BC: Bridging the gap between finite precision and perfect CSIT," *IEEE Transactions on Information Theory*, vol. 64, no. 11, pp. 7208–7217, Nov. 2018.
- [35] A. Motahari, S. Oveis Gharan, and A. Khandani, "Real interference alignment with real numbers," Aug 2009, arXiv:0908.1208.
- [36] A. G. Davoodi and S. A. Jafar, "Aligned image sets under channel uncertainty: Settling conjectures on the collapse of degrees of freedom under finite precision CSIT," *IEEE Transactions on Information Theory*, vol. 62, no. 10, pp. 5603–5618, 2016.
- [37] V. Cadambe and S. Jafar, "Interference Alignment and the Degrees of Freedom of the K user Interference Channel," *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3425–3441, Aug. 2008.
- [38] S. Jafar, "Interference Alignment: A New Look at Signal Dimensions in a Communication Network," in *Foundations and Trends in Communication and Information Theory*, 2011, pp. 1–136.
- [39] A. Gholami Davoodi and S. A. Jafar, "Sum-set inequalities for aligned image sets: Instruments for robust GDoF bounds," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6458–6487, 2020.