Coded matrix computation with gradient coding

Kyungrak Son
Institute of New Media and Communications,
Seoul National University,
Seoul 08826, South Korea,
kyungrakson@snu.ac.kr

Aditya Ramamoorthy
Dept. of Electrical and Computer Eng.,
Iowa State University,
Ames, IA 50011, U.S.A.,
adityar@iastate.edu

2023 IEEE International Symposium on Information Theory (ISIT) | 978-1-6654-7554-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISIT54713.2023.10206996

***Abstract*—Polynomial based approaches, such as the Mat-Dot and entangled polynomial codes (EPC) have been used extensively within coded matrix computations to obtain schemes with good recovery thresholds. However, these schemes are well-recognized to suffer from poor numerical stability in decoding. Moreover, the encoding process in these schemes involves linearly combining a large number of input submatrices, i.e., the encoding weight is high. For the practically relevant case of sparse input matrices, this can have the undesirable effect of significantly increasing the worker node computation time. In this work, we propose a generalization of the EPC scheme by combining the idea of gradient coding along with the basic EPC encoding. Our technique allows us to reduce the weight of the encoding and arrive at schemes that exhibit much better numerical stability; this is achieved at the expense of a worse threshold. By appropriately setting parameters in our scheme, we recover several well-known schemes in the literature. Simulation results show that our scheme provides excellent numerical stability and fast computation speed (for sparse input matrices) as compared to EPC and Mat-Dot codes.**

I. Introduction

Large scale matrix computations are at the heart of various machine learning and optimization problems. In many of these problems, the size of the underlying matrices requires the usage of distributed computing, where the overall job is divided into smaller tasks that can be executed in parallel over multiple workers. However, straightforward task assignments can result in situations where the job execution time is limited by the speed of the slowest worker. This is especially problematic in cloud computing scenarios where workers are well-recognized to exhibit appreciable variance in computing speeds [1].

Background: The field of coded matrix computation [2]–[6] aims at leveraging ideas from coding theory to improve the overall job execution time within distributed clusters. Given matrices $\mathbf{A} \in \mathbb{R}^{\beta\times\alpha}$ and $\mathbf{B} \in \mathbb{R}^{\beta\times\gamma}$, suppose that we are interested in computing $\mathbf{A}^T\mathbf{B}$. In coded computation, a designated central node performs a block decomposition of $\mathbf{A}$ and $\mathbf{B}$ and assigns encoded submatrices of them to the worker nodes. The task of the worker nodes is now to compute the product of these encoded matrices. For carefully designed schemes, it can be shown that the desired result can be decoded as long as *any* τ worker nodes return their results. Thus, the job execution time is not dominated by slow workers. τ is known as the threshold of the scheme.

More recently, it has been recognized [7]–[14] that there are other metrics that are also of interest within coded matrix computation. The work of [7], [9], [10], [15], [16] has demonstrated that several of the original polynomial-based schemes suffer from the problem of numerical instability (i.e., computation error caused by distributing the computation). In particular, the decoded result in these schemes can be essentially useless even for clusters with thirty nodes or more. Furthermore, in several settings, the input matrices $\mathbf{A}$ and $\mathbf{B}$ are sparse. Note that the encoding process typically combines a number of different submatrices of $\mathbf{A}$ (and $\mathbf{B}$); we refer to this as the encoding weight of the scheme. This encoding can significantly increase the number of non-zero entries in the encoded matrices. This in turn will have the undesired effect of increasing the worker node computation time [9], [17], [18]. Thus, coded computation schemes that have small encoding weights are of interest. Other metrics include how well a given scheme leverages partial computations performed by the worker nodes [9], [12], [15], [17].

Within coded computation, the central node first performs a block-decomposition of $\mathbf{A}^T$ and $\mathbf{B}$ as follows.

$$\mathbf{A}^T = \begin{bmatrix} \mathbf{A}_{0,0}^T & \cdots & \mathbf{A}_{p-1,0}^T \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{0,m-1}^T & \cdots & \mathbf{A}_{p-1,m-1}^T \end{bmatrix}, \text{and}$$
$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{0,0} & \cdots & \mathbf{B}_{0,n-1} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{p-1,0} & \cdots & \mathbf{B}_{p-1,n-1} \end{bmatrix}. \quad (1)$$

Each worker node is allowed to store the equivalent of $1/pm$-fraction of $\mathbf{A}$ and $1/pn$-fraction of $\mathbf{B}$. The overall idea is to encode the submatrices of $\mathbf{A}$ and $\mathbf{B}$ and assign the worker nodes the task of computing the product of these encoded submatrices, such that the central node can decode if enough tasks are completed.

Related Work: In polynomial-based schemes [4], [19], [20], the encoding functions are polynomial evaluation maps. Upon multiplication of the encoded matrices, the desired terms appear as coefficients of certain monomials and the other coefficients are treated as interference. If enough worker nodes

The work of K. Son was supported in part by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(grant NRF-2021R1A6A3A01086690). The work of A. Ramamoorthy was supported in part by the National Science Foundation (NSF) under Grant CCF-1910840 and Grant CCF-2115200.

return their results, there are enough evaluation points so that the polynomial can be interpolated and the desired terms and hence $\mathbf{A}^T\mathbf{B}$ can be recovered.

In particular, the Mat-Dot code [19] applies in the setting when $m = n = 1$ and arbitrary p and has recovery threshold of $2p-1$. The entangled polynomial code (EPC) [4] applies for any m, n and p and has threshold of $pmn+p-1$. The decoding process in both cases requires interpolating polynomials of degree $2p-2$ and $pmn+p-2$ respectively. There have been several works that have examined the case of $p = 1$.

The issue of numerical stability has been examined in several works. For instance, [16] works within a different basis set of polynomials. In [10], the authors presented a technique that exploits the properties of rotation and circulant permutation matrices for improved numerical stability and in [9], [11], the authors used random linear combinations for the encoding. Low weight encodings were considered in [9], [17] that also demonstrated a scheme that continues to have the optimal threshold. Finally, techniques that leverage partial stragglers have also been investigated in several works [9], [12], [15], [17]. We note here that for large values of p, m and n, the numerical instability issue with the Mat-Dot and EP code approaches is especially acute. In addition, as we will see their encoding weights are also high, rendering them unsuitable for sparse input matrices.

Main Contributions:

- In this work, we present a coded computation scheme that allows us to trade-off the interpolation degree of the reconstructed polynomial & encoding weight with the recovery threshold for EP and Mat-Dot codes. By operating on this tradeoff we can arrive at schemes that are significantly more stable numerically and suitable for sparse input matrices. Our schemes proceed by combining the idea of gradient coding (GC) [21] and the structure of the EP codes. We calculate the recovery threshold of our scheme.
- We show that [4] and [22] can be viewed as two extremes of the proposed scheme depending on the choice of parameters. Thus, our proposed scheme is a generalization of these schemes.
- Extensive simulation results corroborate our theoretical findings.

We point out that there is a related work that utilizes GC for coded matrix computations in [22]. However, we note that this paper is different from our paper since [22] focuses more on designing numerically stable GC using binary coefficients and does not analyze the recovery threshold. We discuss this in more detail in Sections II-C and III.

Notation: For integers a, b, the notation $a|b$ denotes that a divides b. For a set of vectors $\mathcal{V}$, $\text{span}(\mathcal{V})$ denotes the span of the vectors (i.e., set of all linear combinations of the vectors) in $\mathcal{V}$. If $\mathbf{A}$ is a set of integers then $\mathbf{A} \mod \ell$ denotes $\mathbf{A}$ with all elements reduced modulo ℓ.

II. Sparsity controlled distributed matrix multiplication with general matrix partitions

Definition 1: Gradient Coding matrix. Let $\mathbf{H}$ be a $\eta \times \eta$ matrix, with its rows denoted $\mathbf{h}_i, i = 0, \ldots, \eta-1$. We say that $\mathbf{H}$ is a gradient coding matrix with parameters η and κ if it has the following properties.

(i) It has cyclically shifted rows and each row has $\kappa+1$ non-zero entries. Let $\mathcal{I}_i = \{i, i+1, \ldots, i+\kappa\} \mod \eta$. Row $\mathbf{h}_i = [h_{i,0}\ h_{i,1}\ \ldots\ h_{i,\eta-1}]$ is such that $h_{i,j} \neq 0$, if and only if $j \in \mathcal{I}_i$.

(ii) The all-ones row vector is contained in the span of any $\eta-\kappa$ rows of $\mathbf{H}$, i.e., for $J \subset \{0, \ldots, \eta-1\}$ with $|J| = \eta-\kappa$ we have,

$$\mathbb{1}_{1\times\eta} \in \text{span}(\{\mathbf{h}_i | i \in J\}). \tag{2}$$

A. Motivating example

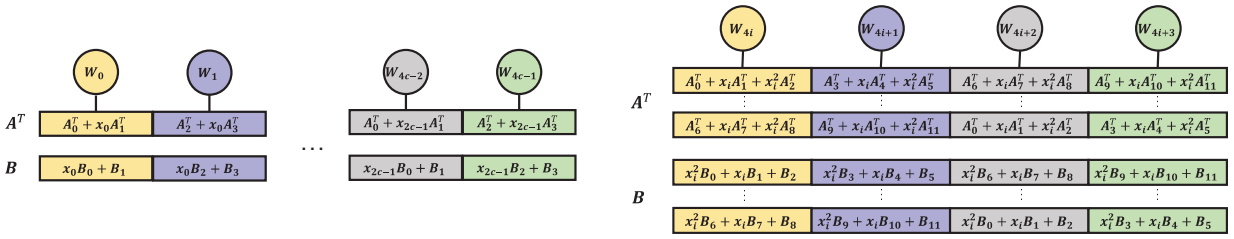
Example 1: Suppose there are $N = 4c$ $(c \geq 5)$ workers. Henceforth, let γ_A and γ_B denote the storage fraction of matrices $\mathbf{A}$ and $\mathbf{B}$. We assume that each worker can store the equivalent of $\gamma_A = \gamma_B = 1/4$ fractions of matrices $\mathbf{A} \in \mathbb{C}^{\beta\times\alpha}$ and $\mathbf{B} \in \mathbb{C}^{\beta\times\gamma}$, respectively. The MatDot code [19] where $\mathbf{A}^T$ and $\mathbf{B}$ are decomposed into four block-columns is applicable here $(m = n = 1, p = 4)$ and is resilient to $N-7$ stragglers. In this approach the encoded $\mathbf{A}$ and $\mathbf{B}$ submatrices involve linear combinations of all the respective submatrices, and decoding requires interpolating a polynomial of degree 6.

Now suppose that we are interested in a scheme where weight of the encoding matrices (both $\mathbf{A}$ and $\mathbf{B}$) is two. In this case, a simple technique is to work with two independent Mat-Dot schemes each with $(m = n = 1, p' = 2)$. We first partition $\mathbf{A}^T$ and $\mathbf{B}$ as $\mathbf{A}^T = [\mathbf{A}_0^T \cdots \mathbf{A}_3^T]$ and $\mathbf{B} = [\mathbf{B}_0 \cdots \mathbf{B}_3]^T$. Then, we divide the workers into $2c$ groups where each group consists of two workers such that the $w = 0, 1$-th worker of the group stores

$$\bar{\mathbf{A}}^T(w, x_i) = \sum_{l=0}^{1} x_i^l \mathbf{A}_{2w+l}^T \quad \text{and} \quad \bar{\mathbf{B}}(w, x_i) = \sum_{l=0}^{1} x_i^{1-l} \mathbf{B}_{2w+l}.$$

as illustrated in Fig. 1(a). The value of x_i is fixed for a group. It is not hard to see that the recovery threshold of above scheme is $2c+3$. Since the product of $\bar{\mathbf{A}}^T(w, x_i)$ and $\bar{\mathbf{B}}(w, x_i)$ yields a degree-2 polynomial, we can decode as long as we obtain three evaluations of each of the two relevant polynomials corresponding to $w = 0, 1$. Thus, we cannot decode when we have all the results, e.g., of the polynomial for $w = 0$ from all of $2c$ groups and the result of the polynomial for $w = 1$ from at most two groups. Thus, the recovery threshold becomes $2c + 3$.

The situation differs when we are interested in schemes where the encoding weight, e.g., is three. In this case, the encoding weight does not divide $p = 4$. Thus, a simple scheme as the one discussed above cannot be found in a straightforward manner. Instead, consider the following scheme. We partition $\mathbf{A}^T$ and $\mathbf{B}$ into 12 submatrices denoted



(a) A simple scheme with two independent MatDot codes.

(b) A group $\mathcal{G}_i$ of the proposed scheme.

Fig. 1. Task assignment in Example 1. There are $N = 4c$ workers for distributed matrix computation with storage size $\gamma_A = \gamma_B = 1/4$. In Fig. 1(b), each worker has three encoded $\mathbf{A}$ and $\mathbf{B}$ assignments. Owing to space limitations, we use vertical dots to denote the missing encoded assignments.

$$\mathbf{A}^T = \begin{bmatrix} \mathbf{A}_0^T & \cdots & \mathbf{A}_{11}^T \end{bmatrix}, \text{ and } \mathbf{B} = \begin{bmatrix} \mathbf{B}_0 \\ \vdots \\ \mathbf{B}_{11} \end{bmatrix}.$$

Since the partitioned submatrices have $1/12$-th the size of matrices $\mathbf{A}$ and $\mathbf{B}$, we can store three of them in each worker while still respecting the storage constraint. Now, consider a group of four workers $\mathcal{G}_i = \{4i, \cdots, 4i+3\}$ for $i = 0, \ldots, c-1$, where the worker $4i + w$ stores

$$\bar{\mathbf{A}}^T(x_i, \tilde{p}) = \sum_{l=0}^{2} x_i^l \mathbf{A}_{3\tilde{p}+l}^T \tag{3}$$

for all $\tilde{p} \in \mathcal{P}_{\mathsf{p}} = \{w, \ldots, w+2\} \mod 4$ (note that the index $\tilde{p}$ depends upon w) and

$$\bar{\mathbf{B}}(x_i, \tilde{p}) = \sum_{l=0}^{2} x_i^{2-l} \mathbf{B}_{3\tilde{p}+l} \tag{4}$$

for all $\tilde{p} \in \mathcal{P}_{\mathsf{p}}$. Here, x_i is the same for all workers in the group. Next, we choose a matrix $\mathbf{H}$ that satisfies (2) with the parameters $\eta = 4$ and $\kappa = 2$ and assign each worker the task of computing

$$\begin{aligned} \mathbf{C}_w(x_i) &= \sum_{\tilde{p}=w}^{w+2} h_{w,\tilde{p}} \bar{\mathbf{A}}^T(x_i, \tilde{p}) \bar{\mathbf{B}}(x_i, \tilde{p}) \\ &\overset{(a)}{=} \sum_{\tilde{p}'=0}^{3} h_{w,\tilde{p}'} \bar{\mathbf{A}}^T(x_i, \tilde{p}') \bar{\mathbf{B}}(x_i, \tilde{p}'), \end{aligned} \tag{5}$$

where $h_{i,j}$ is the i-th row and j-th column of the matrix $\mathbf{H}$. The summation indices in (5) are reduced modulo-4.

Here, (a) holds because of the zeros in the matrix $\mathbf{H}$. When we have at least five groups of workers (i.e., $c \geq 5$), we can prove that this scheme has recovery threshold $= c + 13$.

Lemma 1: The recovery threshold of this scheme is $c + 13$.

Proof: Our overall idea is to show that if the central node can receive at least five evaluations from at least five distinct groups, it can decode the desired result. Towards this end, suppose that the central node receives results from two workers in the group $\mathcal{G}_i$. From Definition 1, there exists $\mathbf{g}^T = [g_0, \cdots, g_3]$ such that it has non-zero entries only corresponding to the two workers that return their results with the property that $\mathbf{g}^T \tilde{\mathbf{h}}_{\tilde{p}} = 1$ for columns $\tilde{\mathbf{h}}_{\tilde{p}}$ of $\mathbf{H}$. Thus,

$$\begin{aligned} &\sum_{w=0}^{3} g_w \mathbf{C}_w(x_i) \\ &= \sum_{w=0}^{3} \sum_{\tilde{p}=0}^{3} g_w h_{w,\tilde{p}} \bar{\mathbf{A}}^T(x_i, \tilde{p}) \bar{\mathbf{B}}(x_i, \tilde{p}) \\ &= \sum_{\tilde{p}=0}^{3} \mathbf{g}^T \tilde{\mathbf{h}}_{\tilde{p}} \bar{\mathbf{A}}^T(x_i, \tilde{p}) \bar{\mathbf{B}}(x_i, \tilde{p}) \\ &= \sum_{\tilde{p}=0}^{3} \bar{\mathbf{A}}^T(x_i, \tilde{p}) \bar{\mathbf{B}}(x_i, \tilde{p}) \qquad (6) \\ &= \sum_{\tilde{p}=0}^{3} \sum_{l_1=0}^{2} \sum_{l_2=0}^{2} x_i^{l_1 - l_2 + 2} \mathbf{A}_{3\tilde{p}+l_1}^T \mathbf{B}_{3\tilde{p}+l_2} \\ &= x_i^2 \sum_{l=0}^{11} \mathbf{A}_l^T \mathbf{B}_l + \text{interference terms} \qquad (7) \end{aligned}$$

Thus, we are able to obtain the useful term $\sum_{l=0}^{11} \mathbf{A}_l^T \mathbf{B}_l$ as the coefficient of x_i^2 in the above polynomial. Furthermore, note that the interference term does not depend on which workers returned their results since we are able to obtain (6) in the decoding process. Since the equation (7) is a polynomial of degree four, we need at least five different interpolation points x_i to obtain the useful term from the equation (7). Thus, obtaining five evaluations from five groups suffices to decode.

To see that the recovery threshold is $c + 13$, we proceed by contradiction. Note that, there are c groups, each of which contains four nodes. It follows that we cannot decode when there are at most four groups where all the nodes return their results and all other groups are such that at most one node returns its result. Thus, we can have at most $4 \times 4 + (c-4) = c + 12$ nodes return their results in this case, i.e., decoding is guaranteed when $c + 13$ workers return their results. ■

B. General k_A, k_B and k_p

We now consider the general case where each worker can store the equivalent of $\gamma_A = 1/k_A k_p$ and $\gamma_B = 1/k_B k_p$ fractions of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. In this case,

the work of [4] considers $k_p \times k_A$ and $k_p \times k_B$ block-decompositions of $\mathbf{A}$ and $\mathbf{B}$ respectively and proposes the EPC scheme with recovery threshold $k_p k_A k_B + k_p - 1$. The encoding weight of the $\mathbf{A}$ and $\mathbf{B}$ matrices is $k_A k_p$ and $k_B k_p$ respectively.

Once again, in this case we are interested in schemes where the encoding weights of the $\mathbf{A}$ and $\mathbf{B}$ is lower and the degree of the polynomial that needs to be interpolated during decoding is lower.

For our scheme, we consider the following scenario. Let m and n be positive integers such that $k_A|m$ and $k_B|n$. Our scheme has another parameter $\Delta_p \leq k_p$ that allows us to tune the weight of the encoding. We set $p = LCM(\Delta_p, k_p)$. As we saw in the motivating example, if $\Delta_p|k_p$, we will see that a simple scheme that essentially divides the overall scheme into $\frac{k_p}{\Delta_p}$ EP codes applies. Thus, for the discussion below we consider the scenario where Δ_p does not divide k_p.

The central node first partitions the matrices $\mathbf{A}^T$ and $\mathbf{B}$ into submatrices as shown in (1). We assume that there are $N = \frac{p}{\Delta_p} \cdot c$ workers, i.e., there are c groups consisting of $\frac{p}{\Delta_p}$ nodes each.

The storage constraints imply that we can store the equivalent of $\frac{pm}{k_p k_A}$ encoded submatrices for $\mathbf{A}^T$ and $\frac{pn}{k_p k_B}$ encoded submatrices for $\mathbf{B}$ in each worker. Thus, we consider a worker group of $\frac{p}{\Delta_p}$ workers $\mathcal{G}_i = \{\frac{p}{\Delta_p} i, \cdots, \frac{p}{\Delta_p}(i+1) - 1\}$ for $i = 0, \ldots, c-1$, where the $\frac{p}{\Delta_p} i + w$-th worker stores

$$\bar{\mathbf{A}}^T(x_i, \tilde{p}, \tilde{m}) = \sum_{l=0}^{\Delta_p - 1} \sum_{s=0}^{k_A - 1} x_i^{l+s\Delta_p} \mathbf{A}^T_{\Delta_p \tilde{p}+l, k_A \tilde{m}+s},$$

for all $\tilde{p} \in \{w, \cdots, w + \frac{p}{k_p} - 1\} \mod \frac{p}{\Delta_p}$ and $\tilde{m} \in \mathcal{P}_{\mathsf{m}} = \{0, \cdots, \frac{m}{k_A} - 1\}$, and

$$\bar{\mathbf{B}}(x_i, \tilde{p}, \tilde{n}) = \sum_{l=0}^{\Delta_p - 1} \sum_{u=0}^{k_B - 1} x_i^{\Delta_p - 1 - l + u\Delta_p k_A} \mathbf{B}_{\Delta_p \tilde{p}+l, k_B \tilde{n}+u},$$

for all $\tilde{p} \in \{w, \cdots, w + \frac{p}{k_p} - 1\} \mod \frac{p}{\Delta_p}$ and $\tilde{n} \in \mathcal{P}_{\mathsf{n}} = \{0, \cdots, \frac{n}{k_B} - 1\}$. Also, x_i is the same for all workers in the group.

Now, we choose a gradient coding matrix (*cf.* Definition 1) $\mathbf{H}$ with parameters $\eta = \frac{p}{\Delta_p}$ and $\kappa = \frac{p}{k_p} - 1$. Then, the w-th worker in the i-th group computes

$$\begin{aligned}\mathbf{C}_w(x_i, \tilde{m}, \tilde{n}) &= \sum_{\tilde{p}=w}^{w+\frac{p}{k_p}-1} h_{w,\tilde{p}} \bar{\mathbf{A}}^T(x_i, \tilde{p}, \tilde{m}) \bar{\mathbf{B}}(x_i, \tilde{p}, \tilde{n}) \\ &= \sum_{\tilde{p}=0}^{\frac{p}{\Delta_p}-1} h_{w,\tilde{p}} \bar{\mathbf{A}}^T(x_i, \tilde{p}, \tilde{m}) \bar{\mathbf{B}}(x_i, \tilde{p}, \tilde{n})\end{aligned}$$

for all $\tilde{m} \in \mathcal{P}_{\mathsf{m}}$ and $\tilde{n} \in \mathcal{P}_{\mathsf{n}}$. The summation indices are reduced modulo $\frac{p}{\Delta_p}$ in the expression above. The last step above holds because of the properties of the GC matrix.

Define $\tau_{\mathsf{GC-EPC}}$ as the recovery threshold of the proposed scheme. The subscript GC-EPC refers to the fact that we combine gradient coding and entangled polynomial coding in this approach. The proof of the following theorem appears in the full version of the paper [23].

Theorem 1: For a given parameter $\Delta_p \leq k_p$, we need at least $c \geq k_A k_B \Delta_p + \Delta_p - 1$ worker groups and the recovery threshold of the scheme is

$$\tau_{\mathsf{GC-EPC}} = \left(\frac{p}{\Delta_p} - \frac{p}{k_p}\right) \cdot c + \frac{p}{k_p} \cdot (k_A k_B \Delta_p + \Delta_p - 2) + 1. \tag{8}$$

Remark 1: From the encoding scheme, we can clearly see that overall polynomial to be interpolated is now of degree $k_A k_B \Delta_p + \Delta_p - 2$ as opposed to $k_A k_B k_p + k_p - 2$ for the EP code. Thus numerical stability improves. Next, the weight of the encoding of $\mathbf{A}$ and $\mathbf{B}$ matrices is $k_A \Delta_p$ and $k_B \Delta_p$ as against $k_A k_p$ and $k_A k_p$ for the EP code, respectively. These benefits come at the cost of a worse recovery threshold.

Example 2: Consider a scenario where $k_A = k_B = 1$ and $k_p = 15$. In this case, the EPC code has a threshold of 29 and the encoding weight of both the $\mathbf{A}$ and $\mathbf{B}$ matrices is 15. We note that interpolating a polynomial of degree 28 will already result in significant numerical issues whereby the decoded result will essentially be useless (see Section III).

For our scheme, suppose that we have $N \geq 5c$ workers and that we set $\Delta_p = 6$. Then, we will choose $p = LCM(6, 15) = 30$. The corresponding threshold will be $3c + 21$, and the encoding weights for both the $\mathbf{A}$ and $\mathbf{B}$ matrices will be 6. We note here that the decoder will only interpolate a polynomial of degree 10 which is much smaller than the Mat-Dot code.

C. Discussions

A comparison of the various performance measures of the proposed scheme and the EPC scheme is summarized in Table I. The recovery threshold and the number of weights of the proposed scheme and the EPC scheme with various system parameters are discussed in Table II. In Table II, $\mathsf{wt}_{\mathsf{epc}}, \mathsf{wt}_{\mathsf{GC-EPC}}$, and $\tau_{\mathsf{epc}}, \tau_{\mathsf{GC-EPC}}$ denotes the encoding weights for EPC scheme and the proposed scheme, and the recovery threshold of the EPC scheme and the proposed scheme respectively. From Table II, we can observe that the threshold of the GC-EPC scheme is higher than the EPC scheme when $\Delta_p < k_p$. However, the encoding weights are lower. Moreover, as discussed shortly in Section III, our scheme is much more numerically stable.

We observe that our scheme reduces to other well-known schemes for specific parameter regimes.

- If $\Delta_p = p = k_p$, the proposed scheme is the same as the EPC scheme [4] and the recovery threshold becomes $k_p k_A k_B + k_p - 1$.
- If $\Delta_p = 1$, $m = n = 1$ and $p = k_p$, the proposed scheme equals to the scheme which just applies GC to uncoded matrices (CMM-1 scheme in [22]) and the recovery threshold becomes $(p-1)c + 1$.
- If $\Delta_p = 1$ and $k_p = p = 1$, the proposed scheme equals to the EPC scheme [4] for $p = 1$ (or CMM-3 scheme in [22]). The recovery threshold becomes $k_A k_B$.

Note that all the three schemes [4], [22], and our proposed GC-EPC scheme have the same computational cost per worker.

	Entangled Polynomial code	Proposed
Recovery threshold	$k_A k_B k_p + k_p - 1$	$\left(\frac{p}{\Delta_p} - \frac{p}{k_p}\right) \cdot c + \frac{p}{k_p} \cdot (k_A k_B \Delta_p + \Delta_p - 2) + 1$
Number of assignments per worker	1	$\frac{pmn}{k_p k_A k_B}$
Computational cost per worker	$O\left(\frac{\alpha\gamma}{k_A k_B} \cdot 2\frac{\beta}{k_p}\right) = O\left(\frac{2\alpha\beta\gamma}{k_p k_A k_B}\right)$	$O\left(\frac{\alpha\gamma}{mn} \cdot 2\frac{\beta}{p} \cdot \frac{pmn}{k_p k_A k_B}\right) = O\left(\frac{2\alpha\beta\gamma}{k_p k_A k_B}\right)$
Encoding weight of $\tilde{\mathbf{A}}$ (or $\tilde{\mathbf{B}}$)	$k_p k_A$ (or $k_p k_B$)	$\Delta_p k_A$ (or $\Delta_p k_B$)

TABLE I
PERFORMANCE COMPARISON OF THE VARIOUS SCHEMES

N	k_A, k_B m, n	k_p	Δ_p	p	τ_{epc}	$\tau_{\mathsf{GC-EPC}}$	$\mathsf{wt}_{\mathsf{epc}}$	$\mathsf{wt}_{\mathsf{GC-EPC}}$
24	1	6	4	12	11	21	6	4
24	1	6	3	6	11	17	6	3
24	1	6	2	6	11	19	6	2
10	1	6	3	6	N/A	10	6	3
64	1	4	3	12	7	29	4	3
64	1	8	3	24	15	53	8	3
64	2	4	3	12	19	56	8	6

TABLE II
SIMPLE COMPARISON OF THE RECOVERY THRESHOLD AND NUMBER OF WEIGHTS WITH THE VARIOUS SYSTEM PARAMETERS

III. SIMULATION RESULT

In this section, we evaluate and compare our proposed schemes with benchmark schemes in terms of two different performance measures.

- First, we measure numerical stability caused by distributing the computations. i.e., the computation error of the reconstructed solution $\hat{\mathbf{C}}$ normalized by the original solution given by

$$\frac{\|\hat{\mathbf{C}} - \mathbf{A}^T\mathbf{B}\|_F}{\|\mathbf{A}^T\mathbf{B}\|_F}.$$

- Second, we compare the average computation time (in seconds), the computation time consumed until receiving the computation results from $\tau_\star, \star = \mathsf{epc}$ or $\mathsf{GC-EPC}$ number of worker nodes, of each scheme.

We compare the performance of the EPC scheme [4] and our proposed scheme; both schemes have the same storage capacity $\gamma_A = 1/k_A k_p$ for the matrix $\mathbf{A}^T$ and $\gamma_B = 1/k_B k_p$ for the matrix $\mathbf{B}$. For both schemes the interpolation points are chosen from points spaced equidistant on the interval $[-1, 1]$.

For the simulation environment, we consider the input matrices $\mathbf{A}$ and $\mathbf{B}$ having the size 5040×5040. We consider the storage parameter as $k_A = 1, k_B = 1$, where each matrix has the 0.01 fraction of nonzero elements (i.e., sparsity parameter $\rho = 0.01$), and $k_p = 14$. The total number of workers is given as $N = 420$.

In Fig. 2, we compare the normalized computation errors and recovery thresholds of various schemes with respect to the sparsity controlling parameter Δ_p. The blue bold lines and the green dashed line represent the computation errors and recovery thresholds, respectively. As expected, we can first observe that the EPC scheme is numerically unstable. On the other hand, the proposed scheme can provide numerical stability while still having straggler resilience. Also, we can observe the tradeoff between the computation error and the recovery threshold. i.e., the computation error of the proposed scheme increases and the recovery threshold decreases as the sparsity controlling parameter Δ_p increases. Finally, we can observe that the recovery threshold of the proposed scheme meets the EPC scheme and GC scheme in extreme cases as we discussed in subsection II-C.

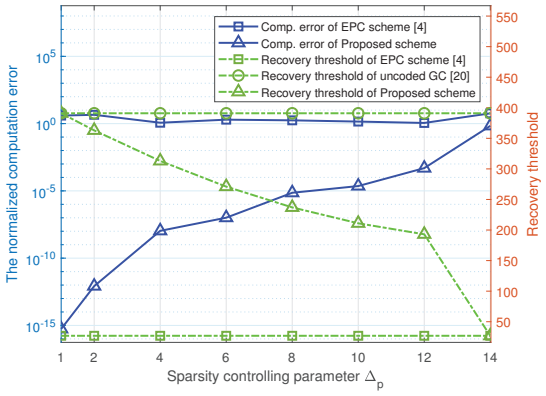


Fig. 2. A plot of the trade-off between computation error and recovery threshold of the various schemes with respect to Δ_p.

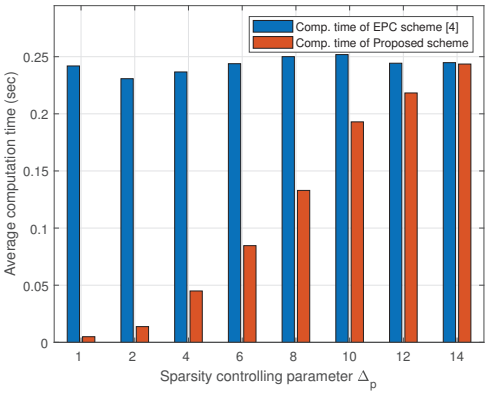


Fig. 3. The average computation time of various schemes with respect to Δ_p.

In Fig. 3, we compare the average computation time of the schemes with respect to Δ_p. We observe that the proposed scheme is faster than the EPC scheme, since the sparsity of the encoded matrices for the proposed scheme is better preserved as compared to that for the EPC scheme.

REFERENCES

[1] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Efficient and robust distributed matrix computations via convolutional coding," *IEEE Trans. Info. Th.*, vol. 67, no. 9, pp. 6266–6282, 2021.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Info. Th.*, vol. 64, no. 3, pp. 1514–1529, 2018.

[3] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Proc. of Adv. in Neur. Inf. Proc. Syst. (NIPS)*, 2017, pp. 4403–4413.

[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Info. Th.*, vol. 66, no. 3, pp. 1920–1933, 2020.

[5] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Proc. of Adv. in Neur. Inf. Proc. Syst. (NIPS)*, 2016, pp. 2100–2108.

[6] A. Ramamoorthy, A. B. Das, and L. Tang, "Straggler-resistant distributed matrix computation via coding theory: Removing a bottleneck in large-scale data processing," *IEEE Sig. Proc. Mag.*, vol. 37, no. 3, pp. 136–145, 2020.

[7] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally decodable matrices for distributed matrix-vector multiplication," in *Proc. of IEEE Intl. Symp. on Info. Th.*, 2019, pp. 1777–1781.

[8] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *Proc. of IEEE Intl. Symp. on Info. Th.*, 2019, pp. 3022–3026.

[9] ——, "Coded sparse matrix computation schemes that leverage partial stragglers," *IEEE Trans. Info. Th.*, vol. 68, no. 6, pp. 4156–4181, 2022.

[10] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," *IEEE Trans. Info. Th.*, vol. 68, no. 4, pp. 2684–2703, 2022.

[11] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao-product codes for numerically-stable distributed matrix multiplication," in *Proc. of Annu. Allerton Conf. Commun. Control Comput*, Sep. 2019, pp. 253–259.

[12] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *Proc. of IEEE Intl. Symp. on Info. Th.*, 2018, pp. 1988–1992.

[13] K. Son, A. Ramamoorthy, and W. Choi, "Distributed matrix multiplication using group algebra for on-device edge computing," *IEEE Sig. Proc. Lett.*, vol. 28, pp. 2097–2101, Oct. 2021.

[14] K. Son and W. Choi, "Distributed matrix multiplication based on frame quantization for straggler mitigation," *IEEE Trans. Signal Process.*, vol. 70, pp. 3058–3073, Jun. 2022.

[15] A. B. Das, L. Tang, and A. Ramamoorthy, "C^3LES : Codes for coded computation that leverage stragglers," in *Proc. of IEEE Info. Th. Workshop*, 2018, pp. 1–5.

[16] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," *IEEE Trans. Info. Th.*, vol. 67, no. 5, pp. 2758–2785, 2021.

[17] A. B. Das and A. Ramamoorthy, "A unified treatment of partial stragglers and sparse matrices in coded matrix computation," *IEEE Jour. on Sel. Area. in Info. Th.*, vol. 3, no. 2, pp. 241–256, 2022.

[18] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *Proc. of Intl. Conf. on Machine Learning (ICML)*, 2018, pp. 5152––5160.

[19] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. Info. Th.*, vol. 66, no. 1, pp. 278–301, 2019.

[20] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the "cubic" barrier," in *Proc. of IEEE Intl. Symp. on Info. Th.*, 2020, pp. 245–250.

[21] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. of Intl. Conf. on Machine Learning (ICML)*, 2017, pp. 3368–3376.

[22] N. Charalambides, H. Mahdavifar, and A. O. Hero III, "Numerically stable binary coded computations," ArXiv preprint, 2021, [Online] Available https://arxiv.org/abs/2109.10484.

[23] K. Son and A. Ramamoorthy, "Coded matrix computation with gradient coding," preprint, 2023, [Online] Available: https://arxiv.org/abs/2304.13685.