Efficient and Robust Distributed Matrix Computations via Convolutional Coding

Anindya Bijoy Das, Aditya Ramamoorthy and Namrata Vaswani
Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA
{abd149,adityar,namrata}@iastate.edu

2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9518108

***Abstract*—Distributed matrix computations are well-recognized to suffer from the problem of stragglers (slow or failed worker nodes). The majority of prior work in this area has presented straggler mitigation strategies that are (i) either sub-optimal in terms of their straggler resilience, or (ii) suffer from numerical problems, i.e., there is a blow-up of round-off errors in the decoded result owing to the high condition numbers of the corresponding decoding matrices. This work introduces a novel solution framework, based on embedding the computations into the structure of a convolutional code, that removes these limitations. Our approach is provably optimal in terms of its straggler resilience, and has excellent numerical robustness which can be theoretically quantified by deriving a computable upper bound on the worst case condition number over all possible decoding matrices. All above claims are backed up by extensive experiments done on the AWS cloud platform.**

I. Introduction

Distributed computing clusters are heavily used in domains such as machine learning where datasets are often so large that they cannot be stored in a single computer. The widespread usage of such clusters presents several advantages over traditional computing paradigms. However, large scale clusters, being heterogeneous in nature, suffer from the issue of stragglers (slow or failed workers). Coded computation introduced in [1] is a technique for structuring distributed computations where the overall job can be resilient to a certain number of stragglers.

The central problem within coded distributed matrix computation can be explained as follows. Suppose that we have large matrices $\mathbf{A} \in \mathbb{R}^{t\times r}, \mathbf{B} \in \mathbb{R}^{t\times w}$ and a vector $\mathbf{x} \in \mathbb{R}^t$. The goal is to either compute $\mathbf{A}^T\mathbf{x}$ (matrix-vector multiplication) or $\mathbf{A}^T\mathbf{B}$ (matrix-matrix multiplication) in a distributed fashion using n worker nodes while being resistant to any s stragglers. Redundancy is introduced in the computation by coding across the submatrices of $\mathbf{A}$ and $\mathbf{B}$ and assigning the worker nodes appropriate computation responsibilities. Several works [1], [2], [3], [4], [5], [6], [7], [8] operate by embedding distributed matrix computations into the structure of erasure codes, where the failed nodes play the role of erasures. A scheme is said to have threshold τ if the desired result can be decoded as long as any τ worker nodes return their results to the master node.

While the threshold is somewhat well understood, the issue of numerical stability has received much less attention. While decoding a real system of equations, errors in the input can get amplified by the condition number (ratio of maximum and minimum singular values) of the associated decoding matrix; hence, a low condition number is critical. For example, the work in [4] uses real Vandermonde matrices for encoding which have very high condition numbers (that are exponential in their size [9]). Some recent works [10], [11], [12] have highlighted the issue of numerical stability in this context.

In this paper we present an efficient and robust scheme for coded matrix computations that is inspired by convolutional codes. Unlike the majority of convolutional codes [13], our codes operate over the reals, and exploit the Vandermonde property of the recovery matrices, where the matrices are defined over a different field (formal Laurent series over $\mathbb{R}$). This naturally allows for simple encoding and decoding in addition to ensuring the threshold properties. We present two classes of codes in this work.

- Our first approach can be decoded with a peeling decoder using only add/subtract operations and has excellent numerical performance when the storage capacity of the nodes is slightly higher than the fundamental lower bound.
- While operating very close to the storage capacity lower bound, we propose an alternative strategy that can provide a "computable" upper bound on the worst case condition number of the recovery matrices. Our work draws novel connections with this problem and the asymptotic analysis of large block Toeplitz matrices [14]. Results in Section IV show that the underlying structure of our codes consistently results in significantly lower worst case condition numbers than [10] and [12].

Owing to space limitations, most of the proofs appear in the longer version of the paper [15].

II. Convolutional Coding for Matrix Computation

We explain our key idea by means of the following example. Consider two row vectors in $\mathbb{R}^q$, $\mathbf{u}_0 = [u_{00}\ u_{01}\ \ldots\ u_{0(q-1)}]$ and $\mathbf{u}_1 = [u_{10}\ u_{11}\ \ldots\ u_{1(q-1)}]$, which can also be represented as polynomials in the *indeterminate* D, $\mathbf{u}_i(D) = \sum_{j=0}^{q-1} u_{ij}D^j$ for $i = 0, 1$. These polynomials can be treated as elements in the ring of formal Laurent series in D [16]. It can be shown that this ring is in fact a field, i.e., each element has a corresponding inverse. Consider the following encoding of $[\mathbf{u}_0(D)\ \ \mathbf{u}_1(D)]$.

$$[\mathbf{c}_0(D)\ \ \mathbf{c}_1(D)\ \ \mathbf{c}_2(D)\ \ \mathbf{c}_3(D)] = [\mathbf{u}_0(D)\ \mathbf{u}_1(D)] \underbrace{\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & D \end{bmatrix}}_{\mathbf{G}(D)}.$$

This work is supported in part by the National Science Foundation (NSF) under Grants CCF-1718470 and CCF-1910840.

It is not too hard to see that the polynomials $\mathbf{u}_0(D)$ and $\mathbf{u}_1(D)$ (equivalently the vectors $\mathbf{u}_0, \mathbf{u}_1$) can be recovered (or "decoded") from any two entries of the vector $[\mathbf{c}_0(D)\ \mathbf{c}_1(D)\ \mathbf{c}_2(D)\ \mathbf{c}_3(D)]$. For instance, suppose that we only receive $\mathbf{c}_2(D)$ and $\mathbf{c}_3(D)$. Notice that

$$\mathbf{c}_2(D) \ = \ \sum_{j=0}^{q-1}(u_{0j}+u_{1j})D^j \quad \text{and}$$

$$\mathbf{c}_3(D) \ = \ u_{00}+\sum_{j=0}^{q-2}(u_{0(j+1)}+u_{1j})D^j+u_{1(q-1)}D^q.$$

Starting with u_{00} from the constant term of $\mathbf{c}_3(D)$, one can iteratively recover each of the coefficients of $\mathbf{u}_0(D)$ and $\mathbf{u}_1(D)$, with only one new variable to recover in each iteration. A similar argument is equally applicable if we consider a different set of two entries from $[\mathbf{c}_0(D)\ \mathbf{c}_1(D)\ \mathbf{c}_2(D)\ \mathbf{c}_3(D)]$. We refer to such a decoding scheme as a "peeling decoder".

A. Proposed matrix-vector multiplication scheme

The above idea can naturally be adapted to the distributed matrix-vector multiplication setting. Consider a system with n workers, s stragglers and threshold $k = n - s$. Suppose that matrix $\mathbf{A}$ is partitioned into $\Delta_A = kq$ block-columns (the choice of q will be discussed shortly). In our work, the presentation follows more naturally if we index the block-columns of $\mathbf{A}$ using two indices, as $\mathbf{A}_{\langle i,j\rangle}, i \in [k], j \in [q]$ (where $[m]$ denotes the set $\{0,\ldots,m-1\}$). Let γ be the storage fraction each worker node, thus each worker stores at most γr columns of matrix $\mathbf{A}$, each having length-t.

Let $\mathbf{U}_i(D) = \sum_{j=0}^{q-1}\mathbf{A}^T_{\langle i,j\rangle}D^j$ for $0 \le i \le k-1$. Furthermore, let $\mathbf{Y}_{k,s}$ denote a $k \times s$ matrix whose (i,j)-th element is $(\mathbf{Y}_{k,s})_{i,j} = (D^j)^i$, for $i \in [k], j \in [s]$, i.e., $\mathbf{Y}_{k,s}$ has the Vandermonde structure. We define

$$\mathbf{G}_{mv}(D) \ = \ \left[\ \mathbf{I}_k \quad \mathbf{Y}_{k,s}(D)\ \right]. \tag{1}$$

Consider the encoding

$$\begin{aligned}&[\mathbf{C}_0(D) \quad \mathbf{C}_1(D) \quad \ldots \quad \mathbf{C}_{n-1}(D)]\\ =\ &[\mathbf{U}_0(D) \quad \mathbf{U}_1(D) \quad \ldots \quad \mathbf{U}_{k-1}(D)]\ \mathbf{G}_{mv}(D).\end{aligned}$$

To arrive at the distributed matrix-vector multiplication scheme, we simply interpret the coefficients of the powers of D in $\mathbf{C}_i(D)$ as the encoded submatrices assigned to worker i. With this assignment, worker i computes the inner product of its assigned matrices and $\mathbf{x}$. The following result shows that $\mathbf{G}_{mv}(D)$ is MDS; (the proof appears at the Appendix in [15]) which implies that $\mathbf{A}^T\mathbf{x}$ can be recovered as long as any k workers complete their tasks.

Corollary 1 (Corollary of upcoming Theorem 1 given in Section II-B)**.** Any $k \times k$ submatrix of $\mathbf{G}_{mv}(D)$ has a determinant which is a non-zero polynomial in D, i.e., it is non-singular.

Analogous to convolutional coding, we call the first k workers the message workers and the last s workers the parity workers. Each of the first k message workers receives q submatrices $\mathbf{A}_{\langle i,j\rangle}, j = 0,1,\ldots,q-1$, each of which is a matrix of size $t \times r/(kq)$. The rest of the s parity workers will receive $\ge q$ such submatrices. The highest exponent of D in the generator matrix $\mathbf{G}_{mv}(D)$ is $(s-1)(k-1)$. Thus, the maximum storage needed by a worker is $q+(s-1)(k-1)$ submatrices. Thus assuming a bound of γ on the storage capacity fraction of any worker, we need $[q+(s-1)(k-1)]\frac{r}{kq} \le \gamma r$ which indicates that

$$q \ge \frac{(s-1)(k-1)}{k(\gamma-\frac{1}{k})}. \tag{2}$$

For instance, if we consider $n = 4$ and $s = 2$ and γ is set to $\frac{5}{8}$, then we can set $q = 4$.

B. Proposed matrix-matrix multiplication scheme

The matrix-matrix multiplication case requires the generalization of the above ideas. Let $\bar{a} = [a_0\ a_1\ \ldots\ a_{s-1}]$ and $\bar{b} = [b_0\ b_1\ \ldots\ b_{k-1}]$ be vectors of non-negative integers such that $0 \le a_0 < a_1 < \cdots < a_{s-1}$ and $0 \le b_0 < b_1 < \cdots < b_{k-1}$. Let $\mathbf{Y}_{\bar{b},\bar{a}}(D)$ denote a $k \times s$ matrix whose (i,j)-th entry is given by

$$[\mathbf{Y}_{\bar{b},\bar{a}}(D)]_{i,j} = (D^{a_j})^{b_i}. \tag{3}$$

Using this matrix, define a generalization of $\mathbf{G}_{mv}(D)$ as

$$\mathbf{G}(D) = \left[\ \mathbf{I}_k \ \middle|\ \mathbf{Y}_{\bar{b},\bar{a}}(D)\ \right]. \tag{4}$$

Observe that we obtain $\mathbf{G}_{mv}(D)$ by setting $a_j = j$ and $b_i = i$, which corresponds to $\mathbf{Y}_{k,s}(D)$. We will design an encoding scheme for matrix-matrix multiplication whose equivalent generator matrix is of the form in (4). Before we explain the design, we show that this matrix also satisfies the MDS property (the proof appears at the Appendix in [15]).

Theorem 1. *Any $k \times k$ submatrix of the generator matrix $\mathbf{G}(D)$ defined in (4) is non-singular.*

While non-singularity by itself does not reveal information about the corresponding condition numbers, Theorem 1 provides a class of schemes with a specific structure that have excellent numerical stability (see Fig. 3 "All Ones" curve) and can be modified and analyzed for condition number using the techniques discussed in Theorem 2 within Section III. The structure of $\mathbf{G}(D)$ in (4) also allows for an efficient peeling decoder; requires only add/subtract operations for decoding.

In the matrix-matrix case, we design generator matrices $\mathbf{G}_A(D)$ of size $k_A \times n$ and $\mathbf{G}_B(D)$ of size $k_B \times n$ such that $s = n - k_Ak_B$. Each worker stores fractions γ_A and γ_B of matrices $\mathbf{A}$ and $\mathbf{B}$ respectively. Let z be a large enough positive integer and let

$$\mathbf{U}^A_i(D) = \sum_{j=0}^{q_A-1}\mathbf{A}^T_{\langle i,j\rangle}D^{zj}, i \in [k_A], \text{ and} \tag{5}$$

$$\mathbf{U}^B_i(D) = \sum_{j=0}^{q_B-1}\mathbf{B}_{\langle i,j\rangle}D^{j}, i \in [k_B]. \tag{6}$$

Furthermore, we let $\mathbf{U}^A(D) = [\mathbf{U}^A_0(D)\ \ldots\ \mathbf{U}^A_{k_A-1}(D)]$ and $\mathbf{U}^B(D) = [\mathbf{U}^B_0(D)\ \ldots\ \mathbf{U}^B_{k_B-1}(D)]$. The final goal

of the master node is to recover all products of the form $\mathbf{A}^T_{\langle i_1,j_1\rangle}\mathbf{B}_{\langle i_2,j_2\rangle}$ for $i_1 \in [k_A], j_1 \in [q_A], i_2 \in [k_B], j_2 \in [q_B]$. Once again by forming

$$[\mathbf{C}^A_0(D)\ \ \mathbf{C}^A_1(D)\ \ \ldots\ \ \mathbf{C}^A_{n-1}(D)] = \mathbf{U}^A(D)\mathbf{G}_A(D), \text{ and}$$
$$[\mathbf{C}^B_0(D)\ \ \mathbf{C}^B_1(D)\ \ \ldots\ \ \mathbf{C}^B_{n-1}(D)] = \mathbf{U}^B(D)\mathbf{G}_B(D),$$

we can represent the assignment of coded submatrices of $\mathbf{A}$ and $\mathbf{B}$ to worker node i by the coefficients of $\mathbf{C}^A_i(D)$ and $\mathbf{C}^B_i(D)$ respectively. Following this step, each worker node computes the pairwise product of each coded submatrix of $\mathbf{A}$ and coded submatrix of $\mathbf{B}$ assigned to it.

The matrices $\mathbf{G}_A(D)$ and $\mathbf{G}_B(D)$ will be picked in such a way so that the pairwise product of each coefficient of $\mathbf{C}^A_i(D)$ and each coefficient of $\mathbf{C}^B_i(D)$ appears in $\mathbf{C}^A_i(D) \times \mathbf{C}^B_i(D)$, i.e., each worker node equivalently computes $\mathbf{C}^A_i(D) \times \mathbf{C}^B_i(D)$. Now using the properties of the Khatri-Rao product, we have

$$\begin{aligned}
&[\mathbf{C}^A_0(D)\ \ \ldots\ \ \mathbf{C}^A_{n-1}(D)] \odot [\mathbf{C}^B_0(D)\ \ \ldots\ \ \mathbf{C}^B_{n-1}(D)] \\
&= \left[\mathbf{U}^A(D)\mathbf{G}_A(D)\right] \odot \left[\mathbf{U}^B(D)\mathbf{G}_B(D)\right] \\
&= \left[\mathbf{U}^A(D) \otimes \mathbf{U}^B(D)\right]\ \left[\mathbf{G}_A(D) \odot \mathbf{G}_B(D)\right]. \qquad (7)
\end{aligned}$$

where $\odot$ and $\otimes$ denotes the Khatri-Rao product [17] and Kronecker Product, respectively.

The key idea at this point is to ensure that $\mathbf{G}_A(D) \odot \mathbf{G}_B(D)$ has the structure of a matrix as in (4). Towards this end, we choose

$$\mathbf{G}_A(D) = \left[\begin{array}{cccc|c} \overbrace{\mathbf{1}_{k_B} & \mathbf{0} & \ldots & \mathbf{0}}^{k_A} \\ \mathbf{0} & \mathbf{1}_{k_B} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ldots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{1}_{k_B} \end{array} \middle| \ \mathbf{Y}_{k_A,s}(D^z)\right],$$

$$\mathbf{G}_B(D) = \left[\ \overbrace{\mathbf{I}_{k_B}\ \ \mathbf{I}_{k_B}\ \ \ldots\ \ \mathbf{I}_{k_B}}^{k_A}\ \middle|\ \mathbf{Y}_{k_B,s}(D)\ \right],$$

where $\mathbf{1}_{k_B}$ is an all-ones row vector of length k_B, and the total number of rows in $\mathbf{G}_A(D)$ and $\mathbf{G}_B(D)$ are k_A and k_B respectively. This implies that

$$\mathbf{G}_A(D) \odot \mathbf{G}_B(D) = [\mathbf{I}_k \ \mid\ \mathbf{Y}_{k_A,s}(D^z) \odot \mathbf{Y}_{k_B,s}(D)] \qquad (8)$$

where $k = k_A k_B$. Lemma 1 (proof appears in [15]) shows that the RHS of (8) has the structure of the matrix in (4).

Lemma 1. The Khatri-Rao product $\mathbf{Y}_{k_A,s}(D^z) \odot \mathbf{Y}_{k_B,s}(D)$ is a matrix in the form of (3), when $z \geq q_B + (s-1)(k_B-1)$.

Lemma 1 explains why Theorem 1 is applicable to the coding scheme used for matrix-matrix multiplication. Thus, this lemma, along with Theorem 1 implies that the proposed convolutional code based matrix-matrix multiplication scheme is MDS. Next, using an approach similar to (2), we can derive

$$q_A \geq \frac{(s-1)(k_A-1)}{k_A(\gamma_A - \frac{1}{k_A})} \quad \text{and} \quad q_B \geq \frac{(s-1)(k_B-1)}{k_B(\gamma_B - \frac{1}{k_B})}.$$

Example 1. Consider the computation of $\mathbf{A}^T\mathbf{B}$ over $n = 6$ workers and $s = 2$ stragglers. Assume that each worker can store/process $\gamma_A = 5/8$ fraction of matrix $\mathbf{A}$ and $\gamma_B = 2/3$ fraction of matrix $\mathbf{B}$. We set $k_A = k_B = 2$, so that $q_A = 4$ and $q_B = 3$. Setting $z = q_B + (s-1)(k_B-1) = 4$, we obtain

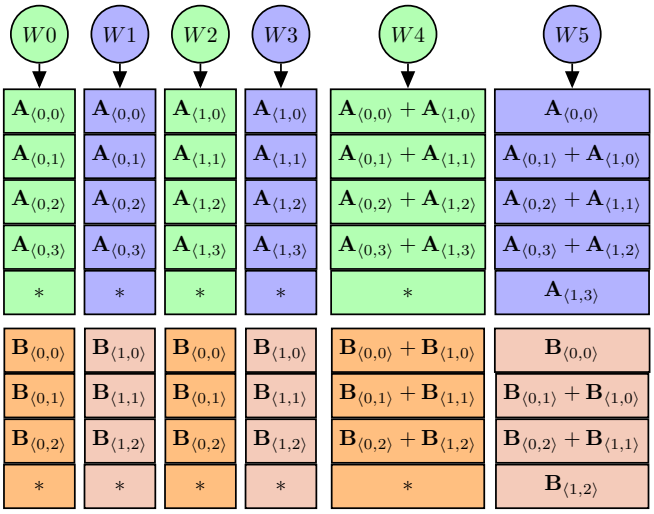

Fig. 1. Matrix-matrix multiplication with $n = 6$ workers and $s = 2$ stragglers with $\gamma_A = \frac{5}{8}$ and $\gamma_B = \frac{2}{3}$.

$$\mathbf{U}^A_i(D) = \sum_{j=0}^{3} \mathbf{A}^T_{\langle i,j\rangle} D^{4j}, \text{ for } i = 0, 1;$$
$$\text{and } \mathbf{U}^B_i(D) = \sum_{j=0}^{2} \mathbf{B}_{\langle i,j\rangle} D^j, \text{ for } i = 0, 1.$$

Furthermore,

$$\mathbf{G}_A(D) = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & D^4 \end{bmatrix} \quad \text{and}$$
$$\mathbf{G}_B(D) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & D \end{bmatrix}.$$

The assignment of jobs to all the workers can be obtained from $[\mathbf{U}^A_0(D) \quad \mathbf{U}^A_1(D)]\,\mathbf{G}_A(D)$ and $[\mathbf{U}^B_0(D) \quad \mathbf{U}^B_1(D)]\,\mathbf{G}_B(D)$. This is shown in Fig. 1.

Remark 1. Our proposed encoding process is very simple and involves only additions at the master node.

C. Effect of q: storage fraction, imbalance in task assignment

Our presented scheme thus far is provably MDS, efficiently decodable and has excellent numerical stability in experiments. Note that our schemes require lower bounds on the value of q which have an inverse dependence on $\gamma - 1/k$. Thus, if one wants to reduce the imbalance between the task assignments to the message nodes and the parity nodes, then q needs to be chosen large enough. It turns out that for large values of q, the worst case condition number of our scheme can be quite large. We present a theoretical treatment of this phenomenon in Section III and discuss techniques for mitigating this effect.

III. Numerical stability analysis

To understand numerical stability, we first introduce a modified encoding scheme and then discuss the matrix representation of the coding ideas described above.

Definition 1 (Randomly scaled generator matrix)**.** Let $\mathbf{R}$ be a $k \times s$ matrix of real numbers. Consider the generator matrix $\mathbf{G}(D)$ defined in (4). Replace $\mathbf{Y}_{\bar{b},\bar{a}}(D)$ by $\mathbf{R} \circ \mathbf{Y}_{\bar{b},\bar{a}}(D)$. Here, $\circ$ denotes the Hadamard product (.* operation in MATLAB).

Note that if we set all entries of the matrix $\mathbf{R}$ to 1, we recover the old generator matrix $\mathbf{G}(D)$ (the "All-Ones" case) mentioned in (4). It is not hard to see that the matrix representation of the transformation induced by the $k \times n$ generator polynomial matrix $\mathbf{G}(D)$ from Definition 1 can be understood as right multiplying a kq-length row vector of input data by the following matrix.

Definition 2 ($\tilde{\mathbf{G}}$: matrix representation of $\mathbf{G}(D)$)**.** We first define a $q \times (q+h)$ shift matrix that takes a q-length row vector and returns a $q+h$-length row vector, where the original vector is shifted to the right by j components. This is the matrix $\tilde{\mathbf{D}}^{h;j} \triangleq \begin{bmatrix}\mathbf{0}_{q\times j} & \mathbf{I}_q & \mathbf{0}_{q\times(h-j)}\end{bmatrix}$. The (i,ℓ)-th block matrix of $\tilde{\mathbf{G}}$ for $\ell = 0,1,\ldots,k-1$ and $i = 0,1,\ldots,k-1$ is

$$(\tilde{\mathbf{G}})_{i,\ell} = \mathbf{I}_q \quad \text{if} \quad i = \ell\,; \quad \text{and} \quad (\tilde{\mathbf{G}})_{i,\ell} = \mathbf{0}_{\mathbf{q}\times\mathbf{q}} \qquad \text{if} \quad i \neq \ell\,;$$

and for $\ell = k+j$, $j = 0,1,\ldots(s-1)$,

$$(\tilde{\mathbf{G}})_{i,\ell} = r_{ij}\tilde{\mathbf{D}}^{a_j b_{k-1}; a_j b_i}.$$

Thus, $\tilde{\mathbf{G}}$ is a $kq \times (nq+\delta)$ matrix where $\delta = b_{k-1}\sum_{j=0}^{s-1} a_j$.

With the above definition, decoding can be understood as inverting the specific $k \times k$ block submatrix of $\tilde{\mathbf{G}}$, denoted $\tilde{\mathbf{G}}_{\mathcal{I}}$ where $\mathcal{I}$ is the set of indices of the k workers that have returned their jobs. Now, since all computing devices are finite precision, matrix multiplications will frequently result in bit overflow/underflow and hence round-off errors. The decoding process amplifies the round-off error by a factor that can at most be as large as the condition number of the decoding matrix [15]. Thus, the numerical stability of our scheme is quantified by the largest condition number over all block submatrices $\tilde{\mathbf{G}}_{\mathcal{I}}$, i.e., by

$$\kappa_{worst} \triangleq \max_{\mathcal{I}\subset[n],|\mathcal{I}|=k} \kappa(\tilde{\mathbf{G}}_{\mathcal{I}}).$$

A. Upper bounding κ_{worst}

Observe that the matrix $\tilde{\mathbf{G}}$, and consequently the decoding submatrix $\tilde{\mathbf{G}}_{\mathcal{I}}$ with $|\mathcal{I}| = k$, has a very specific structure. Because of this, it is possible to show that the matrix $\tilde{\mathbf{G}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}}^T$ is a $k \times k$ block matrix with Toeplitz blocks of size $q \times q$ (see Appendix in [15]). Now, the asymptotics of $\lambda_{\max}(\tilde{\mathbf{G}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}}^T)$ and $\lambda_{\min}(\tilde{\mathbf{G}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}}^T)$ when q is large have been studied in [18]. In particular, Theorem 3 of [18] shows that using Fourier transform ideas, one can bound the eigenvalues of such matrices by computing the minimum (and maximum) of the smallest (and largest) eigenvalues of a much smaller $k \times k$ matrix that is a function of a scalar parameter ω which lies in $[-\pi,\pi]$.

With some abuse of notation, let $\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega})$ represent the matrix obtained by extracting $\mathbf{G}_{\mathcal{I}}(D)$ (from $\mathbf{G}(D)$ in (4)) and then substituting $D = e^{\mathrm{i}\omega}$ (where $\mathrm{i} = \sqrt{-1}$). By adapting the results of [18] (see Appendix in [15] for a detailed description), we have the following theorem.

Theorem 2. For $\mathcal{I} \subset \{0,\ldots,n-1\}$ such that $|\mathcal{I}| = k$,

$$\lim_{q\to\infty} \lambda_{\min}(\tilde{\mathbf{G}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}}^*) = \min_{\omega\in[-\pi,\pi]} \lambda_{\min}[(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))^*];$$
$$\lim_{q\to\infty} \lambda_{\max}(\tilde{\mathbf{G}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}}^*) = \max_{\omega\in[-\pi,\pi]} \lambda_{\max}[(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))^*].$$

Moreover, for any q

$$\lambda_{\max}(\tilde{\mathbf{G}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}}^*) \leq \max_{\omega\in[-\pi,\pi]} \lambda_{\max}[(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))^*];$$
$$\lambda_{\min}(\tilde{\mathbf{G}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}}^*) \geq \min_{\omega\in[-\pi,\pi]} \lambda_{\min}[(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))^*].$$

Theorem 2 shows that we can find an upper bound on the condition number of $\tilde{\mathbf{G}}_{\mathcal{I}}$ based on a scalar optimization over $\omega \in [-\pi,\pi]$. When $\mathbf{R}$ is chosen to be the all-ones matrix, the characterization of Theorem 2 allows us to conclude that when $s > 1$, there exist choices of $\mathcal{I} \subseteq \{0,1,\ldots,n-1\}, |\mathcal{I}| = k$ such that $\tilde{\mathbf{G}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}}^*$ has a minimum eigenvalue that will go to zero as $q \to \infty$. In particular, the corresponding $\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega})$ has repeated columns for $\omega = 0$ (see [15] for an example). Therefore considering a nontrivial scaling of the parity part with a matrix $\mathbf{R}$ is essential for well-conditioned behavior when q is large.

B. Randomly-weighted convolutional coding

Now we show that choosing the matrix $\mathbf{R}$ randomly in Definition 1 results in better numerical stability than the "All-Ones" scheme in the regime of large q. The following result shows that the MDS property continues to hold with probability 1 when the entries are chosen i.i.d. from a continuous distribution. The proof is an easy consequence of Theorem 1 and appears at the Appendix in [15].

Corollary 2. If the entries of the matrix $\mathbf{R}$ are chosen i.i.d. from any continuous-valued probability distribution, then, any $k \times k$ submatrix of the generator matrix mentioned in Definition 1 is non-singular with probability *one*.

We now demonstrate that choosing the matrix $\mathbf{R}$ randomly allows us to upper bound the worst case condition number (over the recovery matrices) even when $q \to \infty$. In the matrix-vector scenario, Theorem 2 suggests the following algorithm for choosing $\mathbf{R}$. Let $\mathcal{I} \subset \{0,\ldots,n-1\}, |\mathcal{I}| = k$ and let $\Omega = \{0, \pm\frac{\pi}{N}, \pm\frac{2\pi}{N}, \ldots, \pm\frac{(N-1)\pi}{N}, \pm\pi\}$ for a large positive integer N denote a fine enough grid of the interval $[-\pi,\pi]$. Let $\kappa_{\mathbf{R}}$ be defined as

$$\max_{\substack{\mathcal{I}\subset\{0,\ldots,n-1\},\\ |\mathcal{I}|=k}} \sqrt{\left(\frac{\max\limits_{\omega\in\Omega}\ \lambda_{\max}[(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))^*]}{\min\limits_{\omega\in\Omega}\ \lambda_{\min}[(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))(\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega}))^*]}\right)}.$$

for any randomly chosen $\mathbf{R}$. Thus $\kappa_{\mathbf{R}}$ indicates the maximum condition number of $\mathbf{G}_{\mathcal{I}}(e^{\mathrm{i}\omega})$ over all $\binom{n}{k}$ choices of $\mathcal{I}$; this is an upper bound on the maximum condition number of $\tilde{\mathbf{G}}_{\mathcal{I}}$. The algorithm repeatedly generates choices of $\mathbf{R}$ and retains the choice that has the lowest value of $\kappa_{\mathbf{R}}$; this is denoted by $\mathbf{R}^\star$. The matrix-matrix case is similar, except that we obtain two random matrices $\mathbf{R}_A^\star$ and $\mathbf{R}_B^\star$, to minimize the worst case condition number of the appropriate submatrices of (8).

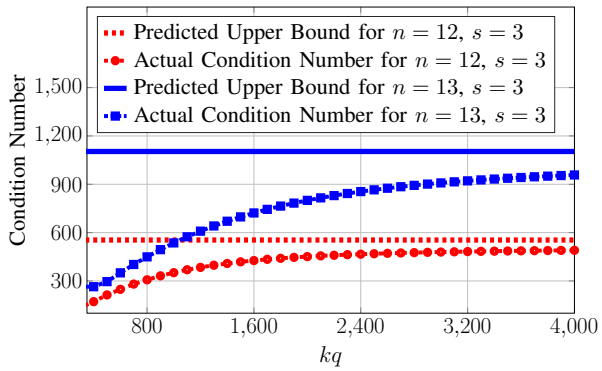

Fig. 2. κ_{worst} for random convolutional code for different n and s.

TABLE I
COMPARISON OF κ_{worst} FOR $n = 18$ AND $s = 3$

METHODS	κ_{worst}
POLYNOMIAL CODE [4]	4.031×10^7
ORTHO-POLY CODE [10]	2.506×10^4
RANDOM KHATRI-RAO CODE[12]	5329.3
CIRCULANT AND ROTATION MATRIX [11]	102
PROPOSED ALL-ONES CONV CODE	**4417.8**
PROPOSED RANDOM CONV CODE	**1829.4**

Example 2. For systems with $n = 12, s = 3$ and $n = 13, s = 3$, we conducted 50 random trials each to find the corresponding $\mathbf{R}^\star$ for matrix vector multiplication. Our algorithm also returns the asymptotic upper bound on $\kappa(\mathbf{R}^\star)$. By sweeping over values of q, we can compute the actual worst-case condition number (κ_{worst}) for each particular chosen value of q. Fig. 2 depicts the upper bound and κ_{worst} for different n and s.

IV. COMPARISONS AND NUMERICAL EXPERIMENTS

We compare our approaches with [4], [10], [11], [12] by numerical experiments which are performed over a cluster in `AWS` (Amazon Web Services); a more exhaustive comparison appears in [15]. A `t2.2xlarge` machine is used as the master node and `t2.small` machines as the slave nodes. Software code for recreating these experiments can be found at [19].

Comparing κ_{worst} and MSE: We choose a system with $n = 18$ workers and $s = 3$ stragglers, and set $\gamma_A = \frac{1}{4}$ and $\gamma_B = \frac{2}{5}$ with $k_A = 5$ and $k_B = 3$, so $k = k_A k_B = n - s = 15$. Table I reports a comparison of the worst-case condition numbers for different approaches in the literature. It can be observed that the work of [4] has much higher condition numbers than both of our proposed schemes (All-ones and Random). Our approaches are also better than the work of [10] and [12] in terms of worst case condition number (κ_{worst}) .

Next in AWS, we choose matrices $\mathbf{A}$ and $\mathbf{B}$ of sizes $15,000 \times 10080$ and $15,000 \times 12000$, respectively. We simulate errors in the worker node computations by adding white Gaussian noise to the calculated submatrix products obtained from the worker nodes and sweeping the range of SNRs. We compare the normalized mean-squared error (MSE) of the different matrix-matrix multiplication methods for their respective worst case scenarios. The corresponding results

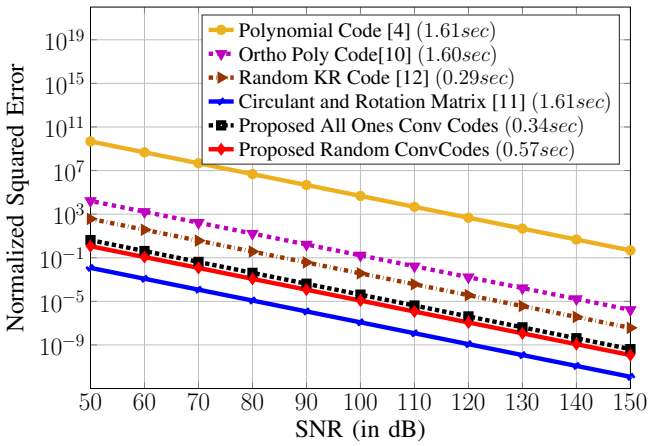

Fig. 3. Comparison of normalized MSE for different approaches (with the corresponding decoding time in the legend).

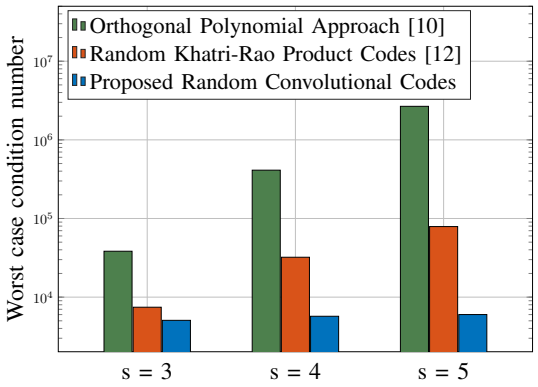

Fig. 4. Comparison of κ_{worst} for matrix-vector multiplication among [10], [12] and our proposed rand. conv. code approach for $n = 20$ with $s = 3, 4$ and 5, where for the proposed method, we used $\gamma = \frac{1}{15}, \frac{1}{14}, \frac{1}{13}$, respectively.

appear in Fig. 3 (for additive Gaussian noise) where we observe that even at $SNR = 70\ dB$, our approach is around 9, 4 and 2 orders of magnitude better than [4], [10] and [12]. The corresponding decoding time is also reported in the legend which shows that the decoding time for our approaches compare quite well with other approaches.

The numerical performance of [11] is better than ours but our scheme has the advantage of much more efficient decoding. The decoding process in [11] requires solving a system of equations whose complexity can be cubic in the size of the unknowns, whereas our proposed 'all-ones' approach can recover the unknowns using a very low-complexity peeling decoder.

Comparing [10], [12] and our approach The RKRP approach in [12] can be considered as specific instance of our random scaling method where the scaling is applied to a trivial all-ones parity matrix, instead of a carefully designed $\mathbf{Y}_{\bar{b},\bar{a}}(D)$. As both approaches are random and pick the best choices, we conducted an experiment where we ran 100 trials for both methods (with $n = 20$ and $s = 3, 4, 5$) and picked the respective best choices (see Fig. 4 for corresponding κ_{worst}). It is clear that the structure imposed in our construction improves the condition number as compared to [12]. Moreover, Fig. 4 also demonstrates the superiority of our approach to [10].

REFERENCES

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

[2] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *IEEE Intl. Symposium on Info. Th.*, 2017, pp. 2418–2422.

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

[4] 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.

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

[7] 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 Meas. and Analysis of Comp. Syst.*, vol. 3, no. 3, pp. 1–40, 2019.

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

[9] V. Pan, "How Bad Are Vandermonde Matrices?" *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 2, pp. 676–694, 2016.

[10] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," in *IEEE Intl. Symposium on Info. Th.*, July 2019, pp. 3017–3021.

[11] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," preprint, 2019, [Online] Available: https://arxiv.org/abs/1910.06515.

[12] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao-Product Codes for Numerically-Stable Distributed Matrix Multiplication," in *57th Annual Conf. on Comm., Control, and Computing (Allerton)*, Sep. 2019, pp. 253–259.

[13] S. Lin and D. J. Costello, *Error Control Coding, 2nd Ed.* Prentice Hall, 2004.

[14] R. M. Gray, "Toeplitz and circulant matrices: A review," *Foundations and Trends® in Comm. and Inf. Th.*, vol. 2, no. 3, pp. 155–239, 2006.

[15] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Efficient and robust distributed matrix computations via convolutional coding," preprint, 2020, [Online] Available: https://arxiv.org/abs/1907.08064.

[16] I. Niven, "Formal power series," *The American Mathematical Monthly*, vol. 76, no. 8, pp. 871–889, 1969.

[17] X.-D. Zhang, *Matrix Analysis and Applications*. Cambridge University Press, 2017.

[18] H. Gazzah, P. A. Regalia, and J.-P. Delmas, "Asymptotic eigenvalue distribution of block Toeplitz matrices and application to blind SIMO channel identification," *IEEE Trans. on Info. Th.*, vol. 47, no. 3, pp. 1243–1251, 2001.

[19] Straggler Mitigation Codes. [Online]. Available: https://github.com/anindyabijoydas/StragglerMitigateConvCodes