Numerically stable coded matrix computations via circulant and rotation matrix embeddings

Aditya Ramamoorthy and Li Tang
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011
{adityar,litang}@iastate.edu

2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9517750

***Abstract*—Polynomial based methods have recently been used in several works for mitigating the effect of stragglers in distributed matrix computations. However, they suffer from serious numerical issues owing to the condition number of the corresponding real Vandermonde-structured recovery matrices. For a system with n worker nodes where s can be stragglers the condition number grows exponentially in n. We present a novel coded computation approach that leverages the properties of circulant permutation and rotation matrices. Our scheme has an optimal recovery threshold and an upper bound on the worst case condition number of our recovery matrices which grows as $\approx O(n^{s+6})$; in the practical scenario where s is a constant, this grows polynomially in n. Our schemes leverage the well-behaved conditioning of complex Vandermonde matrices with parameters on the complex unit circle, while still working with computation over the reals. Exhaustive experimental results demonstrate that our proposed method has condition numbers that are orders of magnitude lower than prior work.**

I. Introduction

In recent years, approaches based on coding theory (referred to as "coded computation") have been effectively used for straggler mitigation. Coded computation offers significant benefits for specific classes of problems such as matrix computations. There have been several works (see, e.g., [1] [2] [3]), that have exploited the correspondence of coded computation with erasure codes (see [4] for a tutorial introduction and relevant references). The matrix computation is embedded into the structure of an underlying erasure code and stragglers are treated as erasures. A scheme is said to have a threshold τ if the master node can decode the intended result (matrix-vector or matrix-matrix multiplication) as long as any τ nodes complete their tasks.

In this work we examine coded computation from the perspective of numerical stability. Erasure coding typically works with operations over finite fields. Solving a linear system of equation over a finite field only requires the corresponding system to be full-rank. However, when operating over the real field, a numerically robust solution can only be obtained if the condition number (ratio of maximum to minimum singular value) [5] of the system of the equations is small. It turns out that several of the well-known coded computation schemes that work by polynomial evaluation/interpolation have serious numerical stability issues owing to the high condition number of corresponding real Vandermonde system of equations. In this work, we present a scheme that leverages the properties of structured matrices such as circulant permutation matrices and rotation matrices for coded computation. These matrices have eigenvalues that lie on the complex unit circle. Our scheme allows us to exploit the significantly better behaved conditioning of complex Vandermonde matrices while still working with computation over the reals. We also present exhaustive comparisons with existing work.

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

II. Problem Formulation

Consider a scenario where the master node has a large $t \times r$ matrix $\mathbf{A} \in \mathbb{R}^{t \times r}$ and either a $t \times 1$ vector $\mathbf{x} \in \mathbb{R}^{t \times 1}$ or a $t \times w$ matrix $\mathbf{B} \in \mathbb{R}^{t \times w}$. The master node wishes to compute $\mathbf{A}^T\mathbf{x}$ or $\mathbf{A}^T\mathbf{B}$ in a distributed manner over n worker nodes in the matrix-vector and matrix-matrix setting respectively. Towards this end, the master node partitions $\mathbf{A}$ (respectively $\mathbf{B}$) into Δ_A (respectively Δ_B) block-columns. Each worker node is assigned $\delta_A \leq \Delta_A$ and $\delta_B \leq \Delta_B$ linearly encoded block-columns of $\mathbf{A}_0, \ldots, \mathbf{A}_{\Delta_A - 1}$ and $\mathbf{B}_0, \ldots, \mathbf{B}_{\Delta_B - 1}$, so that $\delta_A/\Delta_A \leq \gamma_A$ and $\delta_B/\Delta_B \leq \gamma_B$, where γ_A and γ_B represent the storage fraction constraints for $\mathbf{A}$ and $\mathbf{B}$ respectively.

In the matrix-vector case, the i-th worker is assigned encoded submatrices of $\mathbf{A}$ and the vector $\mathbf{x}$ and computes their inner product. In the matrix-matrix case it computes pairwise products of submatrices assigned to it (either all or some subset thereof). We say that a given scheme has *computation threshold* τ if the master node can decode the intended result as long as *any* τ out of n worker nodes complete their jobs.

The overall goal is to (i) design schemes that are resilient to s stragglers (s is a design parameter), while ensuring that the (ii) desired result can be decoded in a efficient manner, and (iii) the decoded result is numerically robust even in the presence of round-off errors and other sources of noise.

Numerical stability is quantified by the condition number $\kappa(\mathbf{M}) = ||\mathbf{M}||||\mathbf{M}^{-1}||$ of a matrix $\mathbf{M}$ where $||\mathbf{M}||$ denotes the maximum singular value of $\mathbf{M}$. For a system of equations $\mathbf{M}\mathbf{y} = \mathbf{z}$, where $\mathbf{z}$ is known and $\mathbf{y}$ is to be determined, if $\kappa(\mathbf{M}) \approx 10^b$, then the decoded result loses approximately b digits of precision [5]. In particular, matrices that are ill-conditioned lead to significant numerical problems when solving linear equations.

III. Numerically Stable Distributed Matrix Computation Schemes

Let $\mathrm{i} = \sqrt{-1}$ and let $[m]$ denote the set $\{0, \ldots, m-1\}$. For a matrix $\mathbf{M}$, $\mathbf{M}(i,j)$ denotes its (i,j)-th entry, whereas $\mathbf{M}_{i,j}$ denotes the (i,j)-th block sub-matrix of $\mathbf{M}$. The notation $\mathbf{M}_1 \otimes \mathbf{M}_2$ denotes the Kronecker product of $\mathbf{M}_1$ and $\mathbf{M}_2$ and the superscript $*$ for a matrix denotes the complex conjugate.

Definition 1. *Rotation matrix.* The 2×2 matrix $\mathbf{R}_\theta$ below is called a rotation matrix.

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \mathbf{Q}\Lambda\mathbf{Q}^*, \text{ where} \quad (1)$$

$$\mathbf{Q} = \frac{1}{\sqrt{2}}\begin{bmatrix} \mathrm{i} & -\mathrm{i} \\ 1 & 1 \end{bmatrix}, \text{ and } \Lambda = \begin{bmatrix} e^{\mathrm{i}\theta} & 0 \\ 0 & e^{-\mathrm{i}\theta} \end{bmatrix}. \quad (2)$$

Definition 2. *Circulant Permutation Matrix.* Let $\mathbf{e}$ be a row vector of length m with $\mathbf{e} = [0\ 1\ 0\ \ldots\ 0]$. Let $\mathbf{P}$ be a $m \times m$ matrix with $\mathbf{e}$ as its first row. The remaining rows are obtained by cyclicly shifting the first row with the shift index equal to the row index. Then $\mathbf{P}^i, i \in [m]$ are said to be circulant permutation matrices. Let $\mathbf{W}$ denote the m-point Discrete Fourier Transform (DFT) matrix, i.e., $\mathbf{W}(i,j) = \frac{1}{\sqrt{m}}\omega_m^{-ij}$ for $i \in [m], j \in [m]$ where $\omega_m = e^{\mathrm{i}\frac{2\pi}{m}}$ denotes the m-th root of unity. Then, it can be shown [6] that $\mathbf{P} = \mathbf{W}\mathrm{diag}(1, \omega_m, \omega_m^2, \ldots, \omega_m^{(m-1)})\mathbf{W}^*$.

Rotation matrices and circulant permutation matrices have the useful property that they are "real" matrices with complex eigenvalues that lie on the unit circle. We use this property extensively in the sequel.

Definition 3. *Vandermonde Matrix.* A $m \times m$ Vandermonde matrix $\mathbf{V}$ with parameters $s_0, s_1, \ldots, s_{m-1} \in \mathbb{C}$ is such that $\mathbf{V}(i,j) = s_j^i, i \in [m], j \in [m]$. If the s_i's are distinct, then $\mathbf{V}$ is nonsingular [7]. In this work, we will also assume that the s_i's are non-zero.

The following facts about $\kappa(\mathbf{V})$ follow from prior work [8].

- *Real Vandermonde matrices.* If $s_i \in \mathbb{R}, i \in [m]$, i.e., if $\mathbf{V}$ is a real Vandermonde matrix, then it is known that its condition number is exponential in m.
- *Complex Vandermonde matrices with parameters "not" on the unit circle.* Suppose that the s_i's are complex and let $s_+ = \max_{i=0}^{m-1} |s_i|$. If $s_+ > 1$ then $\kappa(\mathbf{V})$ is exponential in m. Furthermore, if $1/|s_i| \geq \nu > 1$ for at least $\beta \leq m$ of the m parameters, then $\kappa(\mathbf{V})$ is exponential in β.

Thus, the only scenario where the condition number is somewhat well-behaved is if most or all of the parameters of $\mathbf{V}$ are complex and lie on the unit-circle. In [9], we show the following result which is one of our key technical contributions.

Theorem 1. [9] Consider a $m \times m$ Vandermonde matrix $\mathbf{V}$ where $m < q$ (where q is odd) with distinct parameters $\{s_0, s_1, \ldots, s_{m-1}\} \subset \{1, \omega_q, \omega_q^2, \ldots, \omega_q^{q-1}\}$. Then,

$$\kappa(\mathbf{V}) \leq O(q^{q-m+6}).$$

Remark 1. If $q - m$ is a constant, then $\kappa(\mathbf{V})$ grows only polynomially in q. In the subsequent discussion, we will leverage Theorem 1 extensively.

Illustrative Example: Consider the matrix-vector case where $\Delta_A = 3$ and $\delta_A = 1$. In the polynomial approach, the master node forms $\mathbf{A}(z) = \mathbf{A}_0 + \mathbf{A}_1 z + \mathbf{A}_2 z^2$ and evaluates it at distinct real values $z_1, \ldots, z_n$. The i-th evaluation is sent to the i-th worker node which computes $\mathbf{A}^T(z_i)\mathbf{x}$. The decoding at the master node corresponding to solving a Vandermonde system of equations; when Δ_A is large, the interpolation is numerically unstable [8].

The basic idea of our approach to tackle the numerical stability issue is as follows. We further split each $\mathbf{A}_i$ into two equal sized block-columns. Thus, we now have six block-columns, indexed as $\mathbf{A}_0, \ldots, \mathbf{A}_5$. Consider the 6×2 matrix defined below; its columns are specified by $\mathbf{g}_0$ and $\mathbf{g}_1$.

$$[\mathbf{g}_0\ \mathbf{g}_1] = \begin{bmatrix} \mathbf{I} \\ \mathbf{R}_\theta^i \\ \mathbf{R}_\theta^{2i} \end{bmatrix}.$$

The master node forms "two" encoded matrices for the i-th worker: $\sum_{j=0}^{5} \mathbf{A}_j \mathbf{g}_0(j)$ and $\sum_{j=0}^{5} \mathbf{A}_j \mathbf{g}_1(j)$ (where $\mathbf{g}_i(l)$ denotes the l-th component of the vector $\mathbf{g}_i$); γ_A is still $\frac{1}{3}$. Worker i computes the inner product of these two encoded matrices with $\mathbf{x}$ and sends the result to the master node.

When any three workers i_0, i_1, and i_2 complete their tasks, the decodability and numerical stability of recovering $\mathbf{A}^T\mathbf{x}$ depends on the condition number of the following matrix.

$$\begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} \\ \mathbf{R}_\theta^{i_0} & \mathbf{R}_\theta^{i_1} & \mathbf{R}_\theta^{i_2} \\ \mathbf{R}_\theta^{2i_0} & \mathbf{R}_\theta^{2i_1} & \mathbf{R}_\theta^{2i_2} \end{bmatrix}.$$

Using the eigendecomposition of $\mathbf{R}_\theta$ (*cf.* (2)) the above block matrix can expressed as

$$\begin{bmatrix} \mathbf{Q} & 0 & 0 \\ 0 & \mathbf{Q} & 0 \\ 0 & 0 & \mathbf{Q} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} \\ \Lambda^{i_0} & \Lambda^{i_1} & \Lambda^{i_2} \\ \Lambda^{2i_0} & \Lambda^{2i_1} & \Lambda^{2i_2} \end{bmatrix}}_{\mathbf{\Sigma}} \begin{bmatrix} \mathbf{Q}^* & 0 & 0 \\ 0 & \mathbf{Q}^* & 0 \\ 0 & 0 & \mathbf{Q}^* \end{bmatrix},$$

As the pre- and post-multiplying matrices are unitary, the condition number of the above matrix only depends on the properties of the middle matrix, denoted by $\mathbf{\Sigma}$. Upon appropriate column and row permutations, $\mathbf{\Sigma}$ can be shown equivalent to a block diagonal matrix where each of the blocks is a Vandermonde matrix with parameters on the unit circle [9]. Thus, the matrix is invertible if the corresponding parameters are distinct. Furthermore, even though we use "real computation", the numerical stability of our scheme depends on Vandermonde matrices with parameters on the unit circle. Theorem 1 shows that the condition number of such matrices is much better behaved.

Encoding schemes: In this work our general strategy will be to first partition the matrices $\mathbf{A}$ and $\mathbf{B}$ into $\Delta_A = k_A \ell$ and $\Delta_B = k_B \ell$ block-columns respectively. However, we use two indices to refer to their respective constituent block-columns as this simplifies our later presentation. To avoid confusion,

we use the subscript $\langle i,j\rangle$ to refer to the corresponding (i,j)-th block-columns. In particular $\mathbf{A}_{\langle i,j\rangle}, i \in [k_A], j \in [\ell]$ and $\mathbf{B}_{\langle i,j\rangle}, i \in [k_B], j \in [\ell]$ refer to the (i,j)-th block column of $\mathbf{A}$ and $\mathbf{B}$ respectively, such that

$$\begin{aligned}\mathbf{A} &= [\mathbf{A}_{\langle 0,0\rangle} \ \dots \ \mathbf{A}_{\langle 0,\ell-1\rangle} \mid \dots \mid \mathbf{A}_{\langle k_A-1,0\rangle} \ \dots \ \mathbf{A}_{\langle k_A-1,\ell-1\rangle}],\\ \mathbf{B} &= [\mathbf{B}_{\langle 0,0\rangle} \ \dots \ \mathbf{B}_{\langle 0,\ell-1\rangle} \mid \dots \mid \mathbf{A}_{\langle k_B-1,0\rangle} \ \dots \ \mathbf{A}_{\langle k_B-1,\ell-1\rangle}].\end{aligned} \tag{3}$$

The encoding matrix for $\mathbf{A}$ will be specified by a $k_A\ell \times n\ell$ "generator" matrix $\mathbf{G}$ such that

$$\hat{\mathbf{A}}_{\langle i,j\rangle} = \sum_{\alpha\in[k_A],\beta\in[\ell]} \mathbf{G}(\alpha\ell+\beta, i\ell+j)\mathbf{A}_{\langle\alpha,\beta\rangle} \tag{4}$$

for $i \in [n], j \in [\ell]$. A similar rule will apply for $\mathbf{B}$ and result in encoded matrices $\hat{\mathbf{B}}_{\langle i,j\rangle}$. Thus, in the matrix-vector case worker node i stores $\hat{\mathbf{A}}_{\langle i,j\rangle}$ for $j \in [\ell]$ and $\mathbf{x}$, whereas in the matrix-matrix case it stores $\hat{\mathbf{A}}_{\langle i,j\rangle}$ and $\hat{\mathbf{B}}_{\langle i,j\rangle}$, for $j \in [\ell]$. Therefore, worker i stores $\gamma_A = \ell/\Delta_A = 1/k_A$ and $\gamma_B = \ell/\Delta_B = 1/k_B$ fractions of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. In the matrix-vector case (Section III-A), worker node i computes $\hat{\mathbf{A}}^T_{\langle i,j\rangle}\mathbf{x}$ for $j \in [\ell]$ and transmits them to the master node. In the matrix-matrix case (Section III-B), it computes all ℓ^2 pairwise products $\hat{\mathbf{A}}^T_{\langle i,l_1\rangle}\hat{\mathbf{B}}_{\langle i,l_2\rangle}$ for $l_1 \in [\ell], l_2 \in [\ell]$.

In [9], we also consider the general case when matrices are block decomposed along both rows and columns.

Decoding Scheme: With the above encoding, the decoding process corresponds to solving linear equations. We discuss the matrix-vector case here; the matrix-matrix case is quite similar. In the matrix-vector case, the master node receives $\hat{\mathbf{A}}^T_{\langle i,j\rangle}\mathbf{x}$ of length r/Δ_A for $j \in [\ell]$ from a certain number of worker nodes and wants to decode $\mathbf{A}^T\mathbf{x}$ of length r. Based on our encoding scheme, this can be done by solving a $\Delta_A \times \Delta_A$ linear system of equations r/Δ_A times. The structure of this linear system is inherited from the encoding matrix $\mathbf{G}$. The decoding complexity is $O(\Delta_A^3 + r\Delta_A)$, corresponding to inversion and solving r/Δ_A systems of equations; typically we have $r \gg \Delta_A^2$.

A. Distributed Matrix-Vector Multiplication

1) Rotation Matrix Embedding: Let q be an odd number such that $q \geq n$, $\theta = 2\pi/q$ and $\ell = 2$ (*cf.* block column decomposition in (3)). We choose the generator matrix such that its (i,j)-th block submatrix for $i \in [k_A], j \in [n]$ is given by

$$\mathbf{G}^{rot}_{i,j} = \mathbf{R}^{ji}_\theta \tag{5}$$

Theorem 2. [9] The threshold for the rotation matrix based scheme specified above is k_A. Furthermore, the worst case condition number of the recovery matrices is upper bounded by $O(q^{q-k_A+6})$.

Proof. Follows by generalizing the analysis in the example discussed previously and using Theorem 1; details are in [9]. □

2) Circulant Permutation Embedding: Let $\tilde{q}$ be a *prime* number which is greater than or equal to n. We set $\ell = \tilde{q}-1$, so that $\mathbf{A}$ is sub-divided into $k_A(\tilde{q}-1)$ block-columns as in (3). In this embedding we have an additional step. Specifically, the master node generates the following "precoded" matrices.

$$\mathbf{A}_{\langle i,\tilde{q}-1\rangle} = -\sum_{j=0}^{\tilde{q}-2} \mathbf{A}_{\langle i,j\rangle}, i \in [k_A]. \tag{6}$$

In the subsequent discussion, we work with the set of block-columns $\mathbf{A}_{\langle i,j\rangle}$ for $i \in [k_A], j \in [\tilde{q}]$. The coded submatrices $\hat{\mathbf{A}}_{\langle i,j\rangle}$ for $i \in [n], j \in [\tilde{q}]$ are generated by means of a $k_A\tilde{q}\times n\tilde{q}$ matrix $\mathbf{G}^{circ}$ as follows.

$$\hat{\mathbf{A}}_{\langle i,j\rangle} = \sum_{\alpha\in[k_A],\beta\in[\tilde{q}]} \mathbf{G}^{circ}(\alpha\tilde{q}+\beta, i\tilde{q}+j)\mathbf{A}_{\langle\alpha,\beta\rangle}, \tag{7}$$

where the (i,j)-th block of $\mathbf{G}^{circ}$ can be expressed as

$$\mathbf{G}^{circ}_{i,j} = \mathbf{P}^{ji}, \text{ for } i \in [k_A], j \in [n]. \tag{8}$$

The matrix $\mathbf{P}$ denotes the $\tilde{q} \times \tilde{q}$ circulant permutation matrix introduced in Definition 2. For this scheme the storage fraction $\gamma_A = \tilde{q}/(k_A(\tilde{q}-1))$, i.e., it is slightly higher than $1/k_A$.

Remark 2. The $\hat{\mathbf{A}}_{\langle i,j\rangle}$'s can simply be generated by additions since $\mathbf{G}^{circ}$ is a binary matrix.

Theorem 3. [9] The threshold for the circulant permutation based scheme specified above is k_A. Furthermore, the worst case condition number of the recovery matrices is upper bounded by $O(\tilde{q}^{\tilde{q}-k_A+6})$.

The proof is conceptually similar to the proof of Theorem 2 and relies critically on the fact that all eigenvalues of $\mathbf{P}$ lie on the unit circle and that $\mathbf{P}$ can be diagonalized by the DFT matrix $\mathbf{W}$. The circulant permutation embedding admits an efficient decoding algorithm where the fast Fourier Transform (FFT) plays a key role (details in [9]).

B. Distributed Matrix-Matrix Multiplication

The matrix-matrix case requires the introduction of newer ideas within this overall framework. In this case, a given worker obtains encoded block-columns of both $\mathbf{A}$ and $\mathbf{B}$ and representing the underlying computations is somewhat more involved. Once again we let $\theta = 2\pi/q$, where $q \geq n$ (n is the number of worker nodes) is an odd integer and set $\ell = 2$. Furthermore, let $k_Ak_B < n$. The (i,j)-th blocks of the encoding matrices are given by

$$\begin{aligned}\mathbf{G}^A_{i,j} &= \mathbf{R}^{ji}_\theta, \text{ for } i \in [k_A], j \in [n], \text{ and}\\ \mathbf{G}^B_{i,j} &= \mathbf{R}^{(jk_A)i}_\theta, \text{ for } i \in [k_B], j \in [n].\end{aligned}$$

The master node operates according to the encoding rule discussed previously (*cf.* (4)) for both $\mathbf{A}$ and $\mathbf{B}$. Thus, each worker node stores $\gamma_A = 1/k_A$ and $\gamma_B = 1/k_B$ fraction of $\mathbf{A}$ and $\mathbf{B}$ respectively. The i-th worker node computes the pairwise product of the matrices $\hat{\mathbf{A}}^T_{\langle i,l_1\rangle}\hat{\mathbf{B}}_{\langle i,l_2\rangle}$ for $l_1, l_2 = 0, 1$ and returns the result to the master node. Thus, the master node needs to recover all pair-wise products of the form

$$\mathbf{G}_l^A \otimes \mathbf{G}_l^B = \begin{bmatrix} \mathbf{Q}\mathbf{Q}^* \\ \mathbf{Q}\Lambda^l\mathbf{Q}^* \\ \vdots \\ \mathbf{Q}\Lambda^{l(k_A-1)}\mathbf{Q}^* \end{bmatrix} \otimes \begin{bmatrix} \mathbf{Q}\mathbf{Q}^* \\ \mathbf{Q}\Lambda^{lk_A}\mathbf{Q}^* \\ \vdots \\ \mathbf{Q}\Lambda^{lk_A(k_B-1)}\mathbf{Q}^* \end{bmatrix} = \left((\mathbf{I}_{k_A} \otimes \mathbf{Q}) \begin{bmatrix} \mathbf{I} \\ \Lambda^l \\ \vdots \\ \Lambda^{l(k_A-1)} \end{bmatrix} \begin{bmatrix} \mathbf{Q}^* \end{bmatrix} \right) \otimes \left((\mathbf{I}_{k_B} \otimes \mathbf{Q}) \begin{bmatrix} \mathbf{I} \\ \Lambda^{lk_A} \\ \vdots \\ \Lambda^{lk_A(k_B-1)} \end{bmatrix} \begin{bmatrix} \mathbf{Q}^* \end{bmatrix} \right),$$

$\mathbf{A}_{\langle i,\alpha\rangle}^T \mathbf{B}_{\langle j,\beta\rangle}$ for $i \in [k_A], j \in [k_B]$ and $\alpha, \beta = 0, 1$. Let $\mathbf{Z}$ denote a $1 \times 4k_Ak_B$ block matrix that contains all of these pair-wise products.

Theorem 4. The threshold for the rotation matrix based matrix-matrix multiplication scheme is k_Ak_B. The worst case condition number is bounded by $O(q^{q-k_Ak_B+6})$.

Proof. Let $\tau = k_Ak_B$ and suppose that the workers indexed by $i_0, \ldots, i_{\tau-1}$ complete their tasks. Let $\mathbf{G}_l^A$ denote the l-th block column of $\mathbf{G}^A$ (with similar notation for $\mathbf{G}^B$). For $k_1, k_2 \in \{0, 1\}$ the l-th worker node computes $\hat{\mathbf{A}}_{\langle l,k_1\rangle}^T \hat{\mathbf{B}}_{\langle l,k_2\rangle}$ which can be written as

$$\begin{aligned} &\left(\sum_{\alpha\in[k_A],\beta\in\{0,1\}} \mathbf{G}^A(2\alpha+\beta, 2l+k_1)\mathbf{A}_{\langle\alpha,\beta\rangle}^T \right) \\ &\times \left(\sum_{\alpha\in[k_B],\beta\in\{0,1\}} \mathbf{G}^B(2\alpha+\beta, 2l+k_2)\mathbf{B}_{\langle\alpha,\beta\rangle} \right) \\ &\equiv \mathbf{Z} \cdot (\mathbf{G}^A(:, 2l+k_1) \otimes \mathbf{G}^B(:, 2l+k_2)), \end{aligned}$$

using the properties of the Kronecker product. Based on this, it can be observed that the decodability of $\mathbf{Z}$ at the master node is equivalent to checking whether the following matrix is full-rank.

$$\tilde{\mathbf{G}} = [\mathbf{G}_{i_0}^A \otimes \mathbf{G}_{i_0}^B | \mathbf{G}_{i_1}^A \otimes \mathbf{G}_{i_1}^B | \ldots | \mathbf{G}_{i_{\tau-1}}^A \otimes \mathbf{G}_{i_{\tau-1}}^B].$$

To analyze $\tilde{\mathbf{G}}$, consider the decomposition of $\mathbf{G}_l^A \otimes \mathbf{G}_l^B$, for $l \in [n]$ at the top of the page where the first equality uses the eigenvalue decomposition of $\mathbf{R}_\theta$. Applying the properties of Kronecker products, this can be simplified as

$$\tilde{\mathbf{Q}}_1 \underbrace{\left(\begin{bmatrix} \mathbf{I} \\ \Lambda^l \\ \vdots \\ \Lambda^{l(k_A-1)} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I} \\ \Lambda^{lk_A} \\ \vdots \\ \Lambda^{lk_A(k_B-1)} \end{bmatrix} \right)}_{\mathbf{x}_l} \tilde{\mathbf{Q}}_2$$

where $\tilde{\mathbf{Q}}_1 = (\mathbf{I}_{k_A} \otimes \mathbf{Q}) \otimes (\mathbf{I}_{k_B} \otimes \mathbf{Q})$ and $\tilde{\mathbf{Q}}_2 = \left[\mathbf{Q}^*\right]^{\otimes 2}$. Therefore, we can express

$$\begin{aligned} &[\mathbf{G}_{i_0}^A \otimes \mathbf{G}_{i_0}^B | \mathbf{G}_{i_1}^A \otimes \mathbf{G}_{i_1}^B | \ldots | \mathbf{G}_{i_{\tau-1}}^A \otimes \mathbf{G}_{i_{\tau-1}}^B] \\ &= \tilde{\mathbf{Q}}_1 [\mathbf{X}_{i_0} | \mathbf{X}_{i_1} | \ldots | \mathbf{X}_{i_{\tau-1}}] \begin{bmatrix} \tilde{\mathbf{Q}}_2 & 0 & \ldots & 0 \\ 0 & \tilde{\mathbf{Q}}_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \tilde{\mathbf{Q}}_2 \end{bmatrix}. \end{aligned}$$

Once again, we can conclude that the invertibility and the condition number of $\tilde{\mathbf{G}}$ only depends on $[\mathbf{X}_{i_0} | \mathbf{X}_{i_1} | \ldots | \mathbf{X}_{i_{\tau-1}}]$ as the matrices pre- and post- multiplying it are both unitary. The invertibility of $[\mathbf{X}_{i_0} | \mathbf{X}_{i_1} | \ldots | \mathbf{X}_{i_{\tau-1}}]$ essentially follows from polynomial interpolation; a detailed argument appears in [9]. The argument in [9] also shows that upon appropriate permutation, the matrix $[\mathbf{X}_{i_0} | \mathbf{X}_{i_1} | \ldots | \mathbf{X}_{i_{\tau-1}}]$ can be expressed as a block-diagonal matrix with four blocks each of size $\tau \times \tau$. Each of these blocks is a Vandermonde matrix with parameters from the set $\{1, \omega_q, \omega_q^2, \ldots, \omega_q^{q-1}\}$. Therefore, $[\mathbf{X}_{i_0} | \mathbf{X}_{i_1} | \ldots | \mathbf{X}_{i_{\tau-1}}]$ is non-singular and it follows that the threshold of our scheme is k_Ak_B. An application of Theorem 1 implies that the worst case condition number is at most $O(q^{q-\tau+6})$. □

IV. Comparisons and Numerical Experiments

As discussed previously, the scheme of [2] has condition numbers that are exponential in the recovery threshold τ, whereas our method has a worst case condition number (over the recovery worker node set) that is at most $O(q^{q-\tau+6})$ where q can be chosen to be n if n is odd. This is corroborated by our numerical experiments as well. In Section VII of [13], the authors propose a finite field embedding approach as a potential solution to the numerical issues encountered when operating over the reals. For this purpose the real entries will need to multiplied by appropriate scaling constants and then quantized so that each entry lies with 0 and $p-1$ for a large enough prime p. Following this all computations will be performed within the finite field of order p, i.e., by reducing the computations modulo-p, the product will also be recovered within the finite field and then rescaled to obtain the final real matrix estimate. This technique requires that each $\mathbf{A}_i^T\mathbf{B}_j$ needs to have all its entries within 0 to $p-1$, otherwise there will be errors in the computation. Let α be an upper bound on the absolute value of matrix entries in $\mathbf{A}$ and $\mathbf{B}$. Then, this means that the following dynamic range constraint (DRC),

$$\alpha^2 t \leq p - 1$$

needs to be satisfied. Otherwise, the modulo-p operation will cause arbitrarily large errors.

When working over 64-bit integers, the largest integer is $\approx 10^{19}$. Thus, even if $t < 10^5$, the finite-field embedding method can only support $\alpha \leq 10^7$. Thus, the range is rather limited. Furthermore, considering matrices of limited dynamic range is not a valid assumption. In machine learning scenarios such as deep neural networks, matrix multiplications are applied repeatedly, and the output of one stage serves as the input for the other. Thus, over several iterations the dynamic range of the matrix entries will grow. Consequently, applying this technique will necessarily incur quantization error.

TABLE I: Comparison for $\mathbf{A}^T\mathbf{B}$ matrix-matrix multiplication case with $n = 31$, $k_A = 4$, $k_B = 7$. $\mathbf{A}$ has size 8000×14000, $\mathbf{B}$ has size 8400×14000.

Scheme	γ_A	γ_B	τ	Avg. Cond. Num.	Max. Cond. Num.	Avg. Worker Comp. Time (s)	Dec. Time (s)
Real Vand.	1/4	1/7	28	4.9×10^{12}	2.3×10^{13}	2.132	0.407
Complex Vand.	1/4	1/7	28	27	404	8.421	1.321
Rot. Mat. Embed.	1/4	1/7	28	27	404	2.121	0.408
[10]	1/4	1/7	28	1449	8.3×10^4	2.263	0.412
[11]	1/4	1/7	28	255	5.6×10^4	2.198	0.406
Random Conv. [12]	1/3	1/6	28	-	$\leq 3.4 \times 10^4$	-	-

The most serious limitation of the method comes from the fact that the error in the computation (owing to quantization) is strongly dependent on the actual entries of the $\mathbf{A}$ and $\mathbf{B}$ matrices. In fact, we can generate structured integer matrices $\mathbf{A}$ and $\mathbf{B}$ such that the normalized MSE of their approach is exactly 1.0. Towards this end we first pick the prime $p = 2147483647$ (which is much larger than their publicly available code) so that their method can support higher dynamic range. Next let $r = w = t = 2000$. This implies that α has to be ≤ 1000 by the dynamic range constraint. For $k_A = k_B = 2$, the matrices have the following block decomposition.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} \end{bmatrix}, \text{ and}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} \\ \mathbf{B}_{1,0} & \mathbf{B}_{1,1} \end{bmatrix}.$$

Each $\mathbf{A}_{i,j}$ and $\mathbf{B}_{i,j}$ is a matrix of size 1000×1000, with entries chosen from the following distributions. $\mathbf{A}_{0,0}$, $\mathbf{A}_{0,1}$ are distributed Unif$(0, \ldots, 9999)$ and $\mathbf{A}_{1,0}$, $\mathbf{A}_{1,1}$ distributed Unif$(0, \ldots, 9)$. Next, $\mathbf{B}_{0,0}$, $\mathbf{B}_{0,1}$ are distributed Unif$(0, \ldots, 9)$ and $\mathbf{B}_{1,0}, \mathbf{B}_{1,1}$ distributed Unif$(0, \ldots, 9999)$. In this scenario, the DRC requires us to multiply each matrix by 0.1 and quantize each entry between 0 and 999. Note that this implies that $\mathbf{A}_{1,0}, \mathbf{A}_{1,1}, \mathbf{B}_{0,0}, \mathbf{B}_{0,1}$ are all quantized into zero submatrices since the entry in these four submatrices is less than 10. We label the quantized matrices by the superscript $\tilde{}$. We emphasize that the finite field embedding technique *only* recovers the product of these quantized matrices. However, this product is

$$\tilde{\mathbf{A}}^T\tilde{\mathbf{B}} = \begin{bmatrix} \tilde{\mathbf{A}}_{0,0} & \tilde{\mathbf{A}}_{0,1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}^T \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \tilde{\mathbf{B}}_{1,0} & \tilde{\mathbf{A}}_{1,1} \end{bmatrix} = \mathbf{0}.$$

Thus, the final estimate of the original product $\mathbf{A}^T\mathbf{B}$, denoted as $\widehat{\mathbf{A}^T\mathbf{B}}$ is the all-zeros matrix. This implies that the normalized MSE of their scheme ($\frac{||\mathbf{A}^T\mathbf{B} - \widehat{\mathbf{A}^T\mathbf{B}}||_F}{||\mathbf{A}^T\mathbf{B}||_F}$) is exactly 1.0. We note here that even if we consider other quantization schemes or larger 64-bit primes, one can arrive at adversarial examples such as the ones shown above. For these examples, our methods have a normalized MSE of at most 10^{-27}.

The work of [10] demonstrates an upper bound of $O(q^{2(q-\tau)})$ on the worst case condition number; this grows much faster than our upper bound in the parameter $q - \tau$. In numerical experiments, our worst case condition numbers are much smaller than the work of [10]. Both our scheme and [10] have the optimal threshold when $\mathbf{A}$ and $\mathbf{B}$ are only divided into block-columns (*cf.* Section III)). The comparison when matrices are split across both rows and columns is treated in [9].

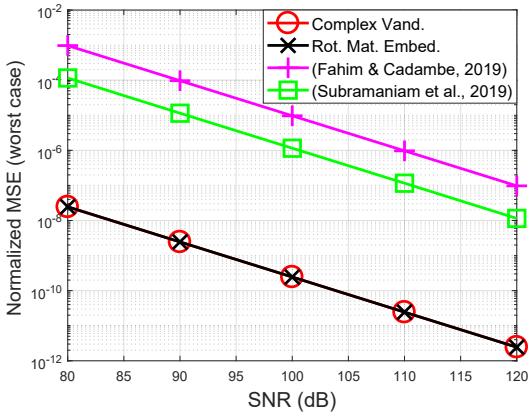

Fig. 1: System with $n = 31$,$k_A = 4$, $k_B = 7$, $\mathbf{A}$ of size 8000×14000 and $\mathbf{B}$ of 8400×14000.

A. Numerical Experiments

We consider a system with $n = 31$ worker nodes and $k_A = 4$ and $k_B = 7$ so that the threshold $\tau = k_A k_B = 28$. Table I shows that the worst case condition number (over worker nodes) of the Rotation Matrix Embedding is about eleven orders of magnitude lower than the Real Vandermonde case. Furthermore, the schemes of [10] and [11] have a worst case condition numbers that are three orders of magnitude and two orders of magnitude higher than our scheme. For both [11] and [12] schemes we performed 200 random trials and picked the scheme with the lowest worst case condition number. For [12], we only report the upper bound on the worst case condition number. Finding the actual worst case recovery set takes a long time. We also include a row corresponding to a complex Vandermonde scheme that operates by using evaluations on the unit-circle. While this gives good numerical performance it results in much higher worker node computation times as shown in the last column of Table I. The normalized MSE simulation [14] was performed by adding white Gaussian noise (of different SNRs) to the encoded matrices. As shown in Figure 1, the normalized MSE of our Rotation Matrix Embedding scheme is much about five orders of magnitude lower than the scheme of [10] and four orders of magnitude better than [11]. The normalized MSE of the Real Vandermonde case is very large so we do not plot it. Since we did not determine the worst case recovery set for [12], we have not included the data and corresponding curves for it.

REFERENCES

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

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

[3] 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 Inf. Theory*, vol. 66, no. 1, pp. 278–301, 2019.

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

[5] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. SIAM:Society for Industrial and Applied Mathematics, 2002.

[6] R. M. Gray, "Toeplitz and circulant matrices: A review," *Foundations and Trends® in Communications and Information Theory*, vol. 2, no. 3, pp. 155–239, 2006.

[7] R. A. Horn and C. R. Johnson, *Topics in matrix analysis*. Cambridge University Press, 1991.

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

[9] 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.

[10] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," *[Online] Available at: https://arxiv.org/abs/1903.08326*, 2019.

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

[12] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Efficient and Robust Distributed Matrix Computations via Convolutional Coding," *[Online] Available at: https://arxiv.org/abs/1907.08064*, 2019.

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

[14] "Repository of numerically stable coded matrix computations via circulant and rotation matrix embeddings," [Online] Available: https://github.com/litangsky/stableCodedComputing.