Systematic Matrix Multiplication Codes

Haewon Jeong*, Yaoqing Yang* and Pulkit Grover
Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—The problem of computing distributed matrix multiplication reliably has been of immense interest for several decades. Recently, it was shown that Polynomial codes achieve the theoretically minimum recovery bandwidth. However, existing constructions for Polynomial codes are nonsystematic, which can impose substantial overhead in distributed computing. In this paper, we propose two different systematic code constructions that achieve the same recovery bandwidth as Polynomial codes. First uses a random coding argument, and the second is polynomial-based, but uses bivariate instead of originally used univariate polynomials. We show that the proposed constructions are communication optimal with high probability.**

I. Introduction

There has been a surge of interest in coded computing [1]–[10], and especially in coded matrix multiplication [11]–[16]. Recently, two optimal points on the trade-off between storage and communication for coded matrix multiplication were identified: MatDot codes [11], that are storage optimal, and Polynomial codes, that are communication optimal [12, Theorem 3]. Recent work also developed a systematic version of MatDot codes [11], but a systematic counterpart for Polynomial codes is so far unavailable. This paper provides a systematic version of Polynomial codes.

Systematic codes can be a more practical coding solution for fully-distributed systems that do not have a master node. Many coded computing strategies proposed in the literature assume a master-worker setup in which all the worker nodes send their computation results back to the master node after each computation. The master node then performs decoding on the gathered result. This centralized formulation is applicable in some cases, e.g., distributed machine learning tasks with a parameter server. However, in many applications, having a single master node in highly-parallel systems introduces unnecessary communication overheads. Acknowledging this issue, recent works (e.g. [2], [5]) have considered masterless setups where even encoding and decoding are fully-distributed.

To understand the advantages of systematic codes in a masterless setup, let us consider computing matrix product $\mathbf{C} = \mathbf{AB}$ and assume the splitting of matrices as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \cdots & \mathbf{B}_m \end{bmatrix}. \tag{1}$$

Note that this splitting is same as Polynomial codes [12] and Product codes [14]. In an uncoded strategy, m^2 workers will compute the product:

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} \mathbf{A_1B_1} & \cdots & \mathbf{A_1B_m} \\ \vdots & \ddots & \vdots \\ \mathbf{A_mB_1} & \cdots & \mathbf{A_mB_m} \end{bmatrix}, \tag{2}$$

where each worker computes one sub-block of $\mathbf{C}$, *i.e.,* $\mathbf{A_iB_j}$. Introducing redundancy for resilience, we add p additional nodes. When failures happen during the computation, any m^2 successful nodes out of $m^2 + p$ nodes can reconstruct the computation output $\mathbf{C}$. In many settings, failures are rare. Under our masterless setting, if we use a non-systematic code, m^2 nodes have to communicate with each other to recover the product $\mathbf{C}$ even when there is no failure, which can be extremely expensive. On the other hand, if we use a systematic code, we do not need any communication to recover $\mathbf{C}$ when all m^2 systematic node – nodes that compute $\mathbf{A_iB_j}$'s $(i, j = 1, \cdots, m)$ – are successful. Further, even when there is a failure among the systematic workers, recovering a failed node only requires communication from m^2 nodes to the failed node. This has a smaller communication complexity than all m^2 nodes communicating with each other. Lastly, encoding systematic codes is more communication efficient since we only have to encode p additional nodes as compared to encoding all $m^2 + p$ nodes in non-systematic codes.

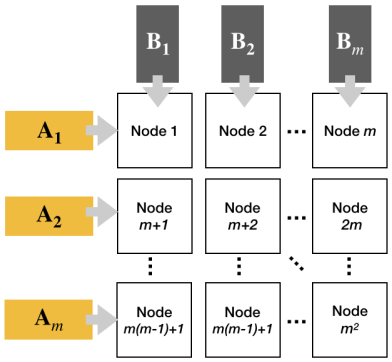

Fig. 1: This diagram shows how we distribute the matrices $\mathbf{A}$ and $\mathbf{B}$ over m^2 systematic nodes. Nodes are placed on a grid in the diagram for the sake of illustration.

II. System Model and Problem Statement

A. System Model

The goal is to compute the matrix product given in (2) where $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{N \times N}$ using $m^2 + p$ distributed nodes. It is assumed that real numbers are stored and processed. The input matrices are split into m sub-blocks of equal dimensions (see (1)). Thus, $\mathbf{A}_i \in \mathbb{R}^{N/m \times N}$ and $\mathbf{B}_i \in \mathbb{R}^{N \times N/m}$. We

* joint first authors.

assume that each node has enough memory to store one sub-block of each $\mathbf{A}$ and $\mathbf{B}$, and compute their product $\mathbf{A}_i\mathbf{B}_j$. Nodes can fail during the computation, and the failures-rate is small. Encoding and decoding are assumed to be failure-free.

B. Problem Statement

Our goal is to construct a systematic Maximum Distance Separable (MDS) code for matrix multiplication split specified in (2), where m^2 nodes compute each sub-block in $\mathbf{C}$ and p redundant nodes perform computation on the encoded matrices. To satisfy the MDS property, we require that the outputs from *any* m^2 nodes out of the total of m^2+p nodes must be sufficient to recover $\mathbf{C}$. We assume that $p < m^2$ as the failure rate is small in our model.

III. Main Result

A. A General Description of Systematic Matrix Multiplication Codes for the Specified Matrix Splitting

We first introduce some notations and set up a framework for systematic matrix multiplication codes under the matrix splitting speified in (1). We denote the "block-vectorized" version of the final matrix $\mathbf{C}$ by:

$$\text{block-vec}(\mathbf{C}) = [\mathbf{A}_1\mathbf{B}_1 \cdots \mathbf{A}_1\mathbf{B}_m \cdots \mathbf{A}_m\mathbf{B}_1 \cdots \mathbf{A}_m\mathbf{B}_m]^T. \tag{3}$$

Let us assume that the matrix blocks $\mathbf{A}_i$'s and $\mathbf{B}_j$'s are scalars for the ease of explanation. We will first explain how we encode input matrices $\mathbf{A}$ and $\mathbf{B}$ and then show how the product $\mathbf{C}$ is encoded as a result.

Systematic encoding matrices for $\mathbf{A}$ and $\mathbf{B}$ are written as:

$$G_{\mathbf{A}} = \left[\begin{array}{ccc} & I_{m\times m} \otimes \mathbf{1}_{m\times 1} & \\ \hline a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{p,1} & \cdots & a_{p,m} \end{array}\right], \tag{4}$$

$$G_{\mathbf{B}} = \left[\begin{array}{ccc} & \mathbf{1}_{m\times 1} \otimes I_{m\times m} & \\ \hline b_{1,1} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{p,1} & \cdots & b_{p,m} \end{array}\right]. \tag{5}$$

We will call the bottom submatrices of these matrices as $P_{\mathbf{A}}$ and $P_{\mathbf{B}}$ respectively, as they are the parity-generating parts. Assuming that $\mathbf{A}_1, \cdots, \mathbf{A}_m, \mathbf{B}_1, \cdots, \mathbf{B}_m$ are scalars, our encoding can be written as:

$$\begin{bmatrix} \widetilde{\mathbf{A}}_1 \\ \vdots \\ \widetilde{\mathbf{A}}_{m^2} \\ \widetilde{\mathbf{A}}_{m^2+1} \\ \vdots \\ \widetilde{\mathbf{A}}_{m^2+p} \end{bmatrix} = G_{\mathbf{A}} \begin{bmatrix} \mathbf{A_1} \\ \vdots \\ \mathbf{A}_m \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_m \\ \vdots \\ \mathbf{A}_m \\ a_{1,1}\mathbf{A}_1 + \cdots + a_{1,m}\mathbf{A}_m \\ \vdots \\ a_{p,1}\mathbf{A}_1 + \cdots + a_{p,m}\mathbf{A}_m \end{bmatrix}, \tag{6}$$

$$\begin{aligned} &\left[\widetilde{\mathbf{B}}_1 \cdots \widetilde{\mathbf{B}}_{m^2} \ \ \widetilde{\mathbf{B}}_{m^2+1} \cdots \widetilde{\mathbf{B}}_{m^2+p}\right] = \left[\mathbf{B}_1 \quad \cdots \quad \mathbf{B}_m\right] G_{\mathbf{B}}^T \\ &= \Big[\mathbf{B}_1 \ \cdots \mathbf{B}_m \cdots \ \mathbf{B}_1 \cdots \mathbf{B}_m \ \ b_{1,1}\mathbf{B}_1 + \cdots + b_{1,m}\mathbf{B}_m \cdots \\ &\ b_{p,1}\mathbf{B}_1 + \cdots + b_{p,m}\mathbf{B}_m\Big]. \end{aligned} \tag{7}$$

$\widetilde{\mathbf{A}}_i$ and $\widetilde{\mathbf{B}}_i$ represent encoded data the node i receives ($i = 1, \cdots, m^2+p$).

The final encoded product can now be written as:

$$\begin{aligned} \widetilde{\mathbf{C}} &= \text{block-vec}(\mathbf{C}) G_{\mathbf{C}} \\ &= \begin{bmatrix} \mathbf{A}_1\mathbf{B}_1 \\ \mathbf{A}_1\mathbf{B}_2 \\ \vdots \\ \mathbf{A}_m\mathbf{B}_m \\ (a_{1,1}\mathbf{A}_1 + \cdots + a_{1,m}\mathbf{A}_m)(b_{1,1}\mathbf{B}_1 + \cdots + b_{1,m}\mathbf{B}_m) \\ \vdots \\ (a_{p,1}\mathbf{A}_1 + \cdots + a_{p,m}\mathbf{A}_m)(b_{p,1}\mathbf{B}_1 + \cdots + b_{p,m}\mathbf{B}_m) \end{bmatrix} \end{aligned} \tag{8}$$

which encodes the block-vectorized form in (3) using an encoding matrix of the form:

$$G_{\mathbf{C}} = \left[\begin{array}{c} I_{m^2\times m^2} \\ \hline P_{\mathbf{A}} \star P_{\mathbf{B}} \end{array}\right]. \tag{9}$$

The $\star$ denotes "row-wise Kronecker product", also known as the Khatri-Rao product [17].

Remark 1. *Remember that we used a simplifying assumption that $\mathbf{A}_i$'s and $\mathbf{B}_j$'s are scalars. To extend this to actual matrices of dimension $N/m \times N$ and $N \times N/m$, we can treat $\mathbf{A}_i, \mathbf{B}_j$'s as elements in the vector space of $\mathbb{R}^{N/m\times N}$*

and $\mathbb{R}^{N\times N/m}$*. Then, we can think of the matrix* $\begin{bmatrix}\mathbf{A}_1\\ \vdots \\ \mathbf{A}_m\end{bmatrix}$ *as an* $m\times 1$ *column vector with each element in* $\mathbb{R}^{N/m\times N}$*. In a similar fashion, the matrix* $\begin{bmatrix}\mathbf{B}_1 & \cdots & \mathbf{B}_m\end{bmatrix}$ *can be regarded as an* $1\times m$ *row vector with each element in* $\mathbb{R}^{N\times N/m}$*. The matrix product* $G_{\mathbf{A}}\cdot\begin{bmatrix}\mathbf{A}_1\\ \vdots \\ \mathbf{A}_m\end{bmatrix}$ *is now a matrix-vector product where the dimension of the matrix is* $(m^2+p)\times m$ *and the length of the vector is* m*. Each element in the matrix is in the field* $\mathbb{R}$ *and each element in the vector is in the vector space* $\mathbb{R}^{N/m\times N}$*. This can be understood as a set of scalar multiplications on the vectors and vector additions.*

While considering the submatrices as vectors is a more intuitive way to understand our construction, we include a "non-vectorized" explanation here. Since our multiplications and additions are performed in a block-wise fashion, the same number should be multiplied to all the elements in the submatrix. E.g., for encoding the first parity node, $a_{1,1}$ should be multiplied with all elements in $\mathbf{A}_1$; $a_{1,2}$ should be multiplied with all elements in $\mathbf{A}_2$, and so on. Since each submatrix $\mathbf{A}_i$ has N/m rows, we have to expand the encoding matrix $G_{\mathbf{A}}$ by N/m as follows:

$$\mathcal{G}_{\mathbf{A}} = G_{\mathbf{A}}\otimes I_{\frac{N}{m}\times\frac{N}{m}}. \tag{10}$$

Now, $\mathcal{G}_{\mathbf{A}}$ is a matrix of dimension $(m^2+p)\frac{N}{m}\times N$, and (6) can be rewritten as:

$$\begin{bmatrix}\widetilde{\mathbf{A}}_1\\ \vdots \\ \widetilde{\mathbf{A}}_{m^2+p}\end{bmatrix} = \mathcal{G}_{\mathbf{A}}\begin{bmatrix}\mathbf{A_1}\\ \vdots \\ \mathbf{A_m}\end{bmatrix}. \tag{11}$$

We can construct different codes by choosing different coefficients in $P_{\mathbf{A}}$ and $P_{\mathbf{B}}$. Our code constructions provided in the following will use this general framework and, we will highlight only how $P_{\mathbf{A}}$ and $P_{\mathbf{B}}$ are constructed.

B. Random Code Construction and Probabilistic Guarantees

Construction 1 (Random Code)**.** *Following the general framework given in* (9)*, all entries in* $P_{\mathbf{A}}$ *and* $P_{\mathbf{B}}$ *are drawn iid from the standard Gaussian distribution* $\mathcal{N}(0,1)$*.*

Theorem 1. *Construction 1 provides a systematic MDS matrix-multiplication code with probability 1, i.e., the results from any* m^2 *out of the overall* m^2+p *nodes are sufficient to reconstruct the final result* $\mathbf{C}$*.*

To prove the theorem, we need two lemmas.

Lemma 2 (Corollary 3, p.319 in [18])**.** *A matrix* G *is an encoding matrix of a systematic MDS code if and only if every square submatrix of the parity generating submatrix* G_P *is non-singular.*

Lemma 3. *If the entries of* $P_{\mathbf{A}}$ *and* $P_{\mathbf{B}}$ *are drawn iid from the standard Gaussian distribution, every square submatrix of the parity generating submatrix* $P_{\mathbf{A}}\star P_{\mathbf{B}}$ *is non-singular with probability* 1*.*

Proof. We will first show that the determinants of any $r\times r$ submatrix ($r\le p$) are non-zero polynomials by mathematical induction. When $r=1$, this is trivial. Now, assume that every $(r-1)\times(r-1)$ submatrix of $P_{\mathbf{A}}\star P_{\mathbf{B}}$ has a non-zero determinant. Let us denote an arbitrary $r\times r$ submatrix as:

$$S=\begin{bmatrix} a_{i_1,j_1}b_{i_1,k_1} & a_{i_1,j_2}b_{i_1,k_2} & \cdots & a_{i_1,j_r}b_{i_1,k_r}\\ a_{i_2,j_1}b_{i_2,k_1} & a_{i_2,j_2}b_{i_2,k_2} & \cdots & a_{i_2,j_r}b_{i_2,k_r}\\ \vdots & \vdots & \ddots & \vdots\\ a_{i_r,j_1}b_{i_r,k_1} & a_{i_r,j_2}b_{i_r,k_2} & \cdots & a_{i_r,j_r}b_{i_r,k_r}\end{bmatrix}. \tag{12}$$

The determinant of this matrix can be written as:

$$\det(S)=a_{i_1,j_1}b_{i_1,k_1}D_1+a_{i_1,j_2}b_{i_1,k_2}D_2+\cdots+a_{i_1,j_r}b_{i_1,k_r}D_r, \tag{13}$$

where D_i is the determinant of the $(r-1)\times(r-1)$ submatrix without the first row and the i-th column of the matrix S, and they are non-zero polynomials due to the induction assumption. Because $(j_1,k_1),(j_2,k_2),\cdots,(j_r,k_r)$ are all distinct, r terms in (13) cannot cancel each other out. Hence, $\det(S)$ is not a zero polynomial.

It is easy to see that the set of $a_{i,j},b_{i,k}$'s in matrix S such that $\det(S)=0$ is a measure-0 subset of the entire space[1]. For a given r, there are $\binom{m^2}{r}\cdot\binom{p}{r}$ possible submatrices. Let us call the set of $a_{i,j},b_{i,k}$'s that makes any square submatrix of $P_{\mathbf{A}}\star P_{\mathbf{B}}$ to have determinant 0, a "bad set". The bad set is a union of $\sum_{r=1}^{p}\binom{m^2}{r}\cdot\binom{p}{r}$ measure-0 subsets. Hence, $P(\text{bad set})=0$ when $a_{i,j},b_{i,k}$'s are chosen randomly from a Gaussian distribution. □

From Lemmas 2 and 3, Theorem 1 follows.

C. Polynomial-based Code Construction

Let us denote a Vandermonde matrix as follows:

$$\text{Vand}_d(u_1,u_2,\cdots u_k)=\begin{bmatrix}1 & u_1 & \cdots & u_1^{d-1}\\ 1 & u_2 & \cdots & u_2^{d-1}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & u_k & \cdots & u_k^{d-1}\end{bmatrix}.$$

Construction 2 (Bivariate Polynomial Code)**.** *Let*

$$\begin{aligned}\mathcal{A}=\text{Vand}_m(\alpha_1,\cdots,\alpha_m),\ \mathcal{B}=\text{Vand}_m(\beta_1,\cdots,\beta_m),\\ \mathcal{A}_P=\text{Vand}_m(\alpha_{m+1},\cdots,\alpha_{m+p}),\\ \mathcal{B}_P=\text{Vand}_m(\beta_{m+1},\cdots,\beta_{m+p}),\end{aligned} \tag{14}$$

where α_i*'s and* β_i*'s* ($i=1,\cdots,m^2+p$) *drawn iid from the standard Gaussian distribution.*

Following the general framework given in (9)*,* $P_{\mathbf{A}}$ *and* $P_{\mathbf{B}}$ *are constructed as follows:*

$$P_{\mathbf{A}}=\mathcal{A}_P\mathcal{A}^{-1}, P_{\mathbf{B}}=\mathcal{B}_P\mathcal{B}^{-1}. \tag{15}$$

The following lemma explains how Construction 2 is based on polynomials.

[1]Depending on which rows and columns are chosen for the submatrix S, the entire space can be as small as $\mathbb{R}^{r^2}$ and as big as $\mathbb{R}^{2r^2}$.

Lemma 4. *If $\alpha_1, \cdots, \alpha_m$ and $\beta_1, \cdots, \beta_m$ are drawn iid from the standard Gaussian distribution, with probability* 1*, there exists a polynomial of degree $2m-2$, $\mathbf{h}(x,y)$ that satisfies*

$$\mathbf{h}(\alpha_i, \beta_j) = \mathbf{A}_i \mathbf{B}_j, \tag{16}$$

for $i, j = 1, \cdots m$.

Proof. If α_i's are all distinct, we can construct a polynomial $\mathbf{f}(x)$ as follows:

$$\mathbf{f}(x) = \sum_{i=1}^{m} \mathbf{A}_i \prod_{j \neq i} \frac{x - \alpha_j}{\alpha_i - \alpha_j}. \tag{17}$$

Similarly, if β_i's are all distinct, we can construct a polynomial $\mathbf{g}(x)$ as follows:

$$\mathbf{g}(x) = \sum_{i=1}^{m} \mathbf{B}_i \prod_{j \neq i} \frac{x - \beta_j}{\beta_i - \beta_j}. \tag{18}$$

Then, if we let $\mathbf{h}(x,y) = \mathbf{f}(x)\mathbf{g}(y)$, (16) is satisfied. For iid samples from Gaussian distribution, $\Pr(\alpha_i = \alpha_j) = 0$ for $i \neq j$. Hence, the polynomial $\mathbf{h}$ exists with probability 1. □

Since the degree of the polynomials $\mathbf{f}$ and $\mathbf{g}$ we constructed in (17) and (18) is $m-1$, let us write them as follows:

$$\mathbf{f}(x) = f_0 + f_1 x + \cdots + f_{m-1} x^{m-1}, \tag{19}$$

$$\mathbf{g}(x) = g_0 + g_1 x + \cdots + g_{m-1} x^{m-1}. \tag{20}$$

Because $\mathbf{f}(\alpha_i) = \mathbf{A}_i$ for $i = 1, 2, \ldots, m$, we have:

$$\begin{bmatrix} f_0 \\ \vdots \\ f_{m-1} \end{bmatrix} = \mathcal{A}^{-1} \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}. \tag{21}$$

For the parity nodes, we encode $\mathbf{A}$ and $\mathbf{B}$ using polynomial evaluations $\mathbf{f}(\alpha_i)$ and $\mathbf{g}(\beta_i)$, $i = m+1, \ldots, m+p$, and let each parity node compute:

$$\mathbf{h}(\alpha_i, \beta_i) = \mathbf{f}(\alpha_i)\mathbf{g}(\beta_i). \tag{22}$$

Using this, our encoding matrix can be written as:

$$G_{\mathbf{A}} = \begin{bmatrix} I_{m \times m} \otimes \mathbf{1}_{m \times 1} \\ \hline \mathcal{A}_P \mathcal{A}^{-1} \end{bmatrix}. \tag{23}$$

The bottom submatrix $P_{\mathbf{A}} = \mathcal{A}_P \mathcal{A}^{-1}$ is the result of polynomial encoding at the parity nodes given in (22). $P_{\mathbf{B}}$ can be obtained similarly.

Theorem 5. *Construction 2 provides a systematic MDS matrix-multiplication code with probability 1, i.e., the results from any m^2 out of the overall m^2+p nodes are sufficient to reconstruct the final result* $\mathbf{C}$*.*

Proof. First, notice that if we can reconstruct the coefficients f_i's and g_j's in polynomial $h(x,y)$, we can recover $\mathbf{C}$ by evaluating $h(x,y)$ at $x = \alpha_i, y = \beta_j$ for $i, j = 1, \cdots, m$. Hence, we will prove that we can reconstruct the polynomial $h(x,y)$ from any m^2 nodes with probability 1, *i.e.,* any $m^2 \times m^2$ submatrix of the following matrix is invertible:

$$H = \begin{bmatrix} \mathcal{A} \otimes \mathcal{B} \\ \hline \mathcal{A}_P \star \mathcal{B}_P \end{bmatrix}_{(m^2+p) \times m^2}. \tag{24}$$

Denote an arbitrary $m^2 \times m^2$ square submatrix of H by S. We will show that $\det(S)$ is a non-zero polynomial of the standard Gaussian random variables α_i's and β_j's, and hence $\Pr(\det(S) = 0) = 0$. We will use $\mathbf{0}$ to denote a zero polynomial.

Let us rewrite S as:

$$S = \begin{bmatrix} S_{sys} \\ \hline S_{par} \end{bmatrix},$$

where S_{sys} and S_{par} are from rows of $\mathcal{A} \otimes \mathcal{B}$ and $\mathcal{A}_P \star \mathcal{B}_P$, respectively. Let us denote the number of rows in S_{sys} and S_{par} as σ and ρ.

Case 1: $\sigma = m^2$ and $\rho = 0$. In other words, $S = \mathcal{A} \otimes \mathcal{B}$. Then, from the property of Kronecker product,

$$\begin{aligned} \det(S) &= \det(\mathcal{A})^m \det(\mathcal{B})^m \\ &= \prod_{i \neq j} (\alpha_i - \alpha_j)^m \prod_{i \neq j} (\beta_i - \beta_j)^m, \end{aligned}$$

which is a non-zero polynomial.

Case 2: $1 \leq \rho \leq p$. We will use induction on ρ.

i) $\rho = 1$.

In this case, S_{par} is a row vector of the following form:

$$S_{par} = \left[\text{Vand}_m(\alpha_k) \otimes \text{Vand}_m(\beta_k) \right],$$

where $k > m$. Using this row vector, the determinant can be expanded as follows:

$$\det(S) = \det(S_1) - \beta_k \det(S_2) + \cdots - \alpha_k^{m-1} \beta_k^{m-1} \det(S_{m^2}), \tag{25}$$

where S_i's are submatrices of S excluding the i-th column and the m^2-th row. The signs in (25) assume that m is even, but the proof holds the same for an odd m. Notice that $\det(S_1), \cdots, \det(S_{m^2})$ are polynomials only in α_i's and β_j's for $i, j = 1, \cdots m$, and they do not have any α_k or β_k terms for $k > m$. Hence, $\det(S) = \mathbf{0}$ only when

$$\det(S_1) = \cdots = \det(S_{m^2}) = \mathbf{0}. \tag{26}$$

i.e., when all these are zero polynomials.

Let us denote $I = \{(i,j) | i, j = 1, \cdots, m\}$ and $I(S) \subseteq I$ as a set of indices of α_i, β_j that are included in S_{sys}. In this case, we have only one element in $I \setminus I(S)$ and let us denote the element as $(\tilde{i}, \tilde{j})$. Now, let us define another matrix S' by replacing S_{par} with

$$\left[\text{Vand}_m(\alpha_{\tilde{i}}) \otimes \text{Vand}_m(\beta_{\tilde{j}}) \right].$$

Notice that the matrix S' now consists of m^2 rows of the systematic part. Therefore, from Case 1, we get:

$$\begin{aligned} \det(S') &= \det(S_1) - \beta_{\tilde{j}} \det(S_2) + \cdots - \alpha_{\tilde{i}}^{m-1} \beta_{\tilde{j}}^{m-1} \det(S_{m^2}) \\ &= \prod_{i \neq j} (\alpha_i - \alpha_j)^m \prod_{i \neq j} (\beta_i - \beta_j)^m \neq \mathbf{0}. \end{aligned}$$

This contradicts (26). Thus, $\det(S) \neq 0$.

ii) Let as assume that $\det(S) \neq 0$ for any $\rho \leq k$. Then, showing that this holds for $\rho = k+1$ is similar to what we did for $\rho = 1$.

$$\det(S) = \det(S_1) - \beta_{k+1} \det(S_2) + \cdots - \alpha_{k+1}^{m-1} \beta_{k+1}^{m-1} \det(S_{m^2}). \tag{27}$$

Now, let us assume that $\det(S) = 0$. This implies that (26) holds. Let us choose $(\tilde{i}, \tilde{j}) \in I \setminus I(S)$ and construct S' by replacing the last row with

$$\left[\text{Vand}_m(\alpha_{\tilde{i}}) \otimes \text{Vand}_m(\beta_{\tilde{j}}) \right].$$

Then S' has k rows from $\mathcal{A}_P \star \mathcal{B}_P$ and $m^2 - k$ rows from $\mathcal{A} \otimes \mathcal{B}$. Thus, by inductive assumption,

$$\det(S') = \det(S_1) - \beta_{\tilde{j}} \det(S_2) + \cdots - \alpha_{\tilde{i}}^{m-1} \beta_{\tilde{j}}^{m-1} \det(S_{m^2}) \neq \mathbf{0}.$$

This contradicts (26). Thus $\det(S) \neq 0$. □

D. Why univariate polynomials do not yield systematic MDS codes

Finally, we want to make a remark on why univariate polynomial structure of Polynomial codes [12] does not easily yield a systematic code construction. We follow the univariate polynomial construction in [12] and assume that the matrices $\mathbf{A}$ and $\mathbf{B}$ are encoded using the polynomials $p_{\mathbf{A}}(x)$ and $p_{\mathbf{B}}(x)$:

$$p_{\mathbf{A}}(x) = \sum_{d=1}^{D_A} f_d(\mathbf{A}_1, \ldots, \mathbf{A}_m) x^d, \tag{28}$$

and

$$p_{\mathbf{B}}(x) = \sum_{d=1}^{D_B} g_d(\mathbf{B}_1, \ldots, \mathbf{B}_m) x^d, \tag{29}$$

for some (possibly linear) functions $f_d(\cdot)$'s and $g_d(\cdot)$'s.

Let us assume that we use polynomial $p_{\mathbf{A}}(x)$ and $p_{\mathbf{B}}(x)$ and the first m^2 workers compute $\mathbf{A_i B_j}$'s $(i, j = 1, \cdots, m)$. This implies the following:

$$p_{\mathbf{A}}(\alpha_n) p_{\mathbf{B}}(\alpha_n) = \mathbf{A}_i \mathbf{B}_j, \tag{30}$$

for $n = m \cdot (i-1) + j$. Then, ignoring constant factors, the following should be satisfied:

$$p_{\mathbf{A}}(\alpha_n) = \mathbf{A}_i, \;\; p_{\mathbf{B}}(\alpha_n) = \mathbf{B}_j. \tag{31}$$

This imposes m^2 evaluation points on both $p_{\mathbf{A}}$ and $p_{\mathbf{B}}$. Hence, the degree of the polynomials $p_{\mathbf{A}}$ and $p_{\mathbf{B}}$ should be at least $m^2 - 1$. Their product, $p_{\mathbf{C}}(x) = p_{\mathbf{A}}(x) \cdot p_{\mathbf{B}}(x)$, thus has degree at least $2m^2 - 2$. This makes the recovery threshold $2m^2 - 1$, instead of m^2.

E. Encoding and Decoding Complexity Analysis

For encoding, we have to generate p different linear combinations of $\mathbf{A}_i$'s and $\mathbf{B}_j$'s which has computational complexity of $O(pN^2)$. For decoding, unlike Polynomial codes that can be decoded using fast polynomial interpolation algorithms, we need to invert the encoding matrix in a brute-force way. The computational complexity of inverting an $m^2 \times m^2$ matrix is $O(m^6)$. For recovering one failed node, it will take $O(m^2 \cdot N^2/m^2) = O(N^2)$. In the worst case scenario where p out of the m^2 systematic nodes fail, the total decoding complexity will be $O(m^6 + pN^2)$ complexity.

IV. Discussion and Future Work

Algorithm-Based Fault Tolerance (ABFT) codes [19] follow the same matrix splitting as in (1), and they are systematic. However, ABFT codes are suboptimal for this split in recovery threshold sense (and hence require more redundant nodes for same fault-tolerance): ABFT codes have a recovery threshold of $2(m-1)(m+\sqrt{p})$, p is perfect square to make $\sqrt{p}$ an integer. For any $m > 1$, this is always greater than Polynomial codes recovery threshold, m^2.

Generalized PolyDot codes [2] interpolate between MatDot and Polynomial codes to gracefully tradeoff between storage and communication. Finding systematic versions of Generalized PolyDot codes remains an open question.

References

[1] Y. Yang, P. Grover, and S. Kar, "Coded distributed computing for inverse problems," in *Advances in Neural Information Processing Systems*, 2017, pp. 709–719.

[2] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A unified coded deep neural network training strategy based on generalized polydot codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1585–1589.

[3] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Advances In Neural Information Processing Systems*, 2016.

[4] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," *arXiv preprint arXiv:1802.03475*, 2018.

[5] H. Jeong, T. Low, and P. Grover, "Masterless coded computing: A fully-distributed coded fft algorithm," *Communication, Control, and Computing (Allerton)*, 2018.

[6] Y. Yang, P. Grover, and S. Kar, "Coding for a single sparse inverse problem," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1575–1579.

[7] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *arXiv preprint arXiv:1806.00939*, 2018.

[8] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded mapreduce," in *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*. IEEE, 2015, pp. 964–971.

[9] H. Yang and J. Lee, "Secure distributed computing with straggling servers using polynomial codes," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 141–150, 2019.

[10] N. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1620–1624.

[11] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *arXiv preprint arXiv:1801.10292*, 2018.

[12] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," *arXiv preprint arXiv:1705.10464*, 2017.

[13] ——, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *arXiv preprint arXiv:1801.07487*, 2018.

[14] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, 2017.

[15] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Information Theory (ISIT), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 2418–2422.

[16] T. Jahani-Nezhad and M. A. Maddah-Ali, "Codedsketch: A coding scheme for distributed computation of approximated matrix multiplications," *arXiv preprint arXiv:1812.10460*, 2018.

[17] S. Liu and G. Trenkler, "Hadamard, khatri-rao, kronecker and other matrix products," *Int. J. Inf. Syst. Sci*, vol. 4, no. 1, pp. 160–177, 2008.

[18] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*. Elsevier, 1977.

[19] K. H. Huang and J. Abraham, "Algorithm-Based Fault Tolerance for Matrix Operations," *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 518–528, 1984.