2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)
Allerton Park and Retreat Center
Monticello, IL, USA, October 2-5, 2018

Locally Recoverable Coded Matrix Multiplication

Haewon Jeong, Fangwei Ye, and Pulkit Grover
haewon@cmu.edu, fwye@ie.cuhk.edu.hk, pulkit@cmu.edu

***Abstract*— Repair locality is important to recover from failed nodes in distributed computing especially when communicating all the data to a master node is expensive. Here, building on recent work on coded matrix multiplication, we provide locally recoverable coded matrix multiplication strategies. Leveraging constructions of optimal matrix multiplication codes and optimal locally recoverable (LRC) codes, we provide constructions of LRC Polynomial codes (minimal communication) and LRC MatDot codes (minimal storage).**

I. Introduction

Coded computing is an emerging advance on classical fault-tolerant computing, and has been shown to address stragglers, faults, and errors in distributed computing problems. Of particular interest has been coded matrix multiplication [1]–[8], which is an immensely important problem as it is a basic building block of most algorithms in machine learning and scientific computing.

This paper focuses on introducing locally recoverable property to matrix multiplication codes. Locally recoverable (LRC) codes have been extensively studied for distributed storage as they can reduce the number of node access to repair a failed storage node [9]–[17]. In particular, Gopalan *et al.* [10] proved that the minimum distance of an (n, k, r)-LRC must satisfy the Singleton-type bound:

$$d \leqslant n - k - \left\lceil \frac{k}{r} \right\rceil + 2,$$

which reduces to the classical Singleton bound $d \leqslant n - k + 1$ when $r = k$. Codes satisfying this bound with equality are called optimal LRC codes. Among the constructions of optimal LRC codes, we utilize the celebrated results of [15] where codes are constructed over alphabets growing linearly in the block length.

Having repair locality can be useful in distributed computing for several scenarios:

- When we want to perform consecutive matrix multiplications, e.g., computing the product $\mathbf{D} = \mathbf{ABC}$, we can repair a failed node locally after computing the first product $\mathbf{D}' = \mathbf{AB}$. Then, we carry on the next computation $\mathbf{D} = \mathbf{D}'\mathbf{C}$ without all the nodes sending their intermediate results to the master node.
- In the fully distributed setting, which does not have a powerful master node, we can use a systematic code with locality. Assuming that the fault rate is low and single node failure is the most common scenario, we can recover a failed systematic node by contacting only a few other nodes.

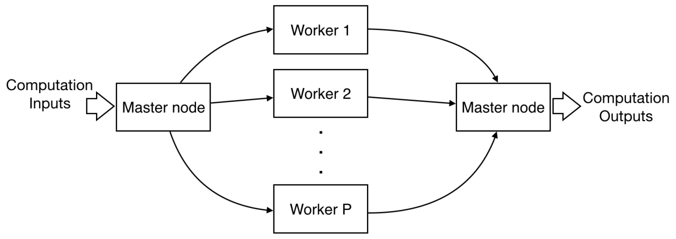

Fig. 1: A master-worker system: a master node first distributes computational inputs to P worker nodes. Worker nodes perform the pre-defined computation on the given inputs then return the computation output back to the master node. The master node performs post-processing if necessary and produces the final computation output.

To the best of our knowledge, this is the first work that proposes locally recoverable matrix multiplication codes. Section II introduces our system model, preliminaries on matrix multiplication codes and LRC codes, and problem statement. Section III provides our main results, which include LRC Polynomial codes, LRC MatDot codes, and the systematic construction of LRC MatDot codes. Here, we leverage novel matrix multiplication codes [1], [3] and optimal LRC codes [15] to obtain locally-recoverable Polynomial codes (which require minimal communication from workers to master node) and locally-recoverable MatDot codes (which are storage optimal). We conclude by discussing potential directions of future work.

II. System Model, Preliminaries, and Problem Statement

A. System Model

Master-worker system: Our system model and problem formulation follows previous works [1], [3], [18]. The system has a master node and P worker nodes; a master node initially has computational inputs and distribute the inputs to worker nodes after some preprocessing if needed (e.g., encoding). Worker nodes receive computation input, perform the given computation, and send the output of the computation back to a master node. We assume that all P worker nodes have the same computational power and storage. A master node, upon receiving the output from all the workers, performs post-processing (e.g., decoding), and obtains the final output of the computation.

Computation Goal: The computation we want to carry out here is matrix multiplication:

$$\mathbf{C} = \mathbf{AB}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are real number matrices of dimension $N \times N$, *i.e.,* $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{N \times N}$.

Storage-constrained worker nodes: We assume that the computation must be distributed due to storage constraints at worker nodes. Specifically, we assume that each worker can only store $1/m$-th fraction of the input matrices, $\mathbf{A}$ and $\mathbf{B}$. In other words, a worker node receives N^2/m symbols of $\mathbf{A}$ and N^2/m symbols of $\mathbf{B}$.

Erasure model: We assume that when a worker node fails, we lose the entire output of the failed node. A node failure can be either a straggler or a node shutdown due to random faults. It is assumed that the node failure is always detected, and thus these failures are modeled as erasures.

Recovery threshold and recovery bandwidth: The recovery threshold is the minimum number of successful workers required by the master node to recover the computation output. We will denote the recovery threshold by K. The recovery bandwidth is the minimum number of symbols to be communicated to the master node to recover the computation output.

B. Coded Matrix Multiplication: Polynomial codes and MatDot codes

For the system model given in Section II-A where a worker node has storage constraint that limits the node to store only $1/m$-th fraction of matrices $\mathbf{A}$ and $\mathbf{B}$, several coding strategies were proposed recently. In this work, we consider two strategies: Polynomial codes which achieve the best recovery bandwidth and MatDot codes which achieve the best recovery threshold. Let us first describe Polynomial codes.

Construction 1. ***Polynomial codes***
The matrix $\mathbf{A}$ is split horizontally into m row-blocks and $\mathbf{B}$ is split vertically into m column-blocks as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}, \quad \mathbf{B} = [\mathbf{B}_1 \ \mathbf{B}_2 \ \ldots \ \mathbf{B}_m], \tag{1}$$

where $\mathbf{A}_i, \mathbf{B}_i$ $(i = 1, \cdots, m)$ are $N/m \times N$ and $N \times N/m$ dimensional submatrices, respectively.

Then, we encode matrix $\mathbf{A}$ and $\mathbf{B}$ using the following polynomials:

$$p_{\mathbf{A}}(x) = \sum_{i=1}^{m} \mathbf{A}_i x^{i-1}, \quad p_{\mathbf{B}}(x) = \sum_{j=1}^{m} \mathbf{B}_i x^{m(j-1)}. \tag{2}$$

A master node distributes encoded matrices, $p_{\mathbf{A}}(\alpha_i), p_{\mathbf{B}}(\alpha_i)$ to the i-th worker node $(i = 1, \cdots, P)$. The i-th worker node then computes the following product at $x = \alpha_i$:

$$p_{\mathbf{C}}(x) = \sum_{i=1}^{m} \sum_{j=1}^{m} \mathbf{A}_i \mathbf{B}_j x^{i-1+m(j-1)}, \tag{3}$$

and send the result to the master node.

Note that the coefficient of $x^{i-1+m(j-1)}$ is $\mathbf{A}_i\mathbf{B}_j$. Since the degree of the polynomial $p_{\mathbf{C}}(x)$ is $m^2 - 1$, once the master node receives the evaluation of $p_{\mathbf{C}}(x)$ at any m^2 distinct points, it can recover the coefficients of $p_{\mathbf{C}}(x)$. Hence, the recovery threshold $K = m^2$. □

Now, we describe MatDot codes.

Construction 2. ***MatDot Codes***
The matrix $\mathbf{A}$ is split vertically into m column-blocks and $\mathbf{B}$ is split horizontally into m row blocks as follows:

$$\mathbf{A} = [\mathbf{A}_1 \ \mathbf{A}_2 \ \ldots \ \mathbf{A}_m], \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_m \end{bmatrix}, \tag{4}$$

where $\mathbf{A}_i, \mathbf{B}_i$ $(i = 1, \cdots, m)$ are $N \times N/m$ and $N/m \times N$ dimensional submatrices, respectively.

Then we encode matrix $\mathbf{A}$ and $\mathbf{B}$ using the following polynomials:

$$p_{\mathbf{A}}(x) = \sum_{i=1}^{m} \mathbf{A}_i x^{i-1}, \quad p_{\mathbf{B}}(x) = \sum_{j=1}^{m} \mathbf{B}_i x^{m-j}. \tag{5}$$

A master node distributes encoded matrices, $p_{\mathbf{A}}(\alpha_i)$ and $p_{\mathbf{B}}(\alpha_i)$ to the i-th worker node $(i = 1, \cdots, P)$. Then the i-th worker node computes the following product at $x = \alpha_i$:

$$p_{\mathbf{C}}(x) = \sum_{i=1}^{m} \sum_{j=1}^{m} \mathbf{A}_i \mathbf{B}_j x^{m-1+(i-j)}, \tag{6}$$

and returns the result to the master node. Note that he coefficient of x^{m-1} in the equation $p_{\mathbf{C}}(x)$ is $\mathbf{C} = \sum_{i=1}^{m} A_i B_i$. Since the degree of the polynomial $p_{\mathbf{C}}(x)$ is $2m - 2$, once the master node receives the evaluation of $p_{\mathbf{C}}(x)$ at any $2m - 1$ distinct points, it can recover the coefficients of $p_{\mathbf{C}}(x)$. Hence, the recovery threshold $K = 2m - 1$. This was proven to be the optimal recovery threshold for the given storage constraint – a worker node can store $1/m$-th fraction of each input matrix [4]. □

C. Locally Recoverable Codes

We say that a code $\mathcal{C}$ has locality r if every symbol of the codeword can be recovered from a subset of r other symbols. In [10], a Singleton-type bound was derived on the maximum distance of LRC codes with locality r.

Theorem 1. *Let $\mathcal{C}$ be an (n, k, r) LRC code. Then the minimum distance of $\mathcal{C}$ satisfies:*

$$d \leqslant n - k - \lceil \frac{k}{r} \rceil + 2. \tag{7}$$

Comparing this with the (n, k) MDS code without locality which has $d = n - k + 1$, we can see that the overhead of having locality is at least $\lceil \frac{k}{r} \rceil - 1$. In this work, we use a family of optimal LRC codes presented in [15] that achieves the equality in (7).

Construction 3. ***Optimal (n,k,r) LRC code [15]***
Let $\boldsymbol{a} \in \mathbb{F}_q^k$ be a message vector and let us re-index $\boldsymbol{a}$ as $\boldsymbol{a} = (a_{ij}, i = 1, \cdots, r; j = 1, \cdots \frac{k}{r})$. For simplicity, we will

assume that r divides k here. Then, the encoding polynomial is defined as:

$$f_{\boldsymbol{a}}(x)=\sum_{i=1}^{r}\sum_{j=1}^{\frac{k}{r}} a_{ij}x^{i-1}g(x)^{j-1}. \tag{8}$$

Let $\mathcal{A}=\{\alpha_1,\cdots,\alpha_n\}$ be a subset of $\mathbb{F}_q$ ($q\geqslant n$). The codeword is the evaluation of the polynomial $f_{\boldsymbol{a}}$ at $\alpha_1,\cdots,\alpha_n$: $\boldsymbol{c}=(f_{\boldsymbol{a}}(\alpha),\alpha\in\mathcal{A})$. A core of this construction is choosing a good polynomial $g(x)$ which satisfies the following:

i) $\deg(g)=r+1$.
ii) There exists a partition of $\mathcal{A}$, $\mathcal{A}=\mathcal{A}_1\bigcup\cdots\bigcup\mathcal{A}_{\frac{n}{r+1}}$, where $|\mathcal{A}_i|=r+1$ such that g is constant on each set $\mathcal{A}_i$. In other words, for all $\alpha,\alpha'\in\mathcal{A}_i$, $g(\alpha)=g(\alpha')$.

First, note that by choosing g with degree $r+1$, the degree of $f_{\boldsymbol{a}}$ becomes $(r+1)\cdot(k/r-1)+r-1=k+r/k-2$. Hence, the distance $d=n-k-\frac{k}{r}+2$. This satisfies the equality in (7).

Now, let us see how choosing such a g guarantees locality r. Let us denote $\mathcal{A}_1=\{\alpha_1,\cdots,\alpha_{r+1}\}$. Without loss of generality, let us assume that $f_{\boldsymbol{a}}(\alpha_1)$ is lost. We want to recover $f_{\boldsymbol{a}}(\alpha_1)$ using r other symbols. Note that, by the second condition, $g(\alpha_1)=g(\alpha_2)=\cdots=g(\alpha_{r+1})=\gamma$. Then, $f_{\boldsymbol{a}}(x)$ at $\alpha_1,\cdots,\alpha_{r+1}$ can be represented as:

$$\begin{aligned}f_{\boldsymbol{a}}(\alpha_l)&=\sum_{i=1}^{r}\sum_{j=1}^{\frac{k}{r}} a_{ij}\alpha_l^{i-1}\gamma^{j-1}\\&=\sum_{i=1}^{r}\Big(\sum_{j=1}^{\frac{k}{r}} a_{ij}\gamma^{j-1}\Big)\alpha_l^{i-1}\\&=\sum_{i=1}^{r}\psi_i\alpha_l^{i-1}\quad(l=1,\cdots,r+1)\end{aligned}$$

Since this is degree-$(r-1)$ polynomial in α_l, the coefficients, ψ_i's can be recovered from evaluation at r points: $f_{\boldsymbol{a}}(\alpha_2),\cdots,f_{\boldsymbol{a}}(\alpha_{r+1})$. Then, we can recover $f_{\boldsymbol{a}}(\alpha_1)$ by computing:

$$f_{\boldsymbol{a}}(\alpha_1)=\sum_{i=1}^{r}\psi_i\alpha_1^{i-1}.$$

□

D. Problem Statement

Under the system model given in Section II-A, we want to give a coding strategy for computing $\mathbf{C}=\mathbf{AB}$ with locality r. More specifically, we want to construct locally recoverable Polynomial codes with locality r and locally recoverable MatDot codes with locality r.

III. Locally Recoverable Matrix Multiplication Codes

In this section, we demonstrate how we can make Polynomial codes and MatDot codes locally recoverable by applying the ideas of optimal LRC codes [15]. In each subsection, we will start by giving a simple example of the construction to provide a better understanding.

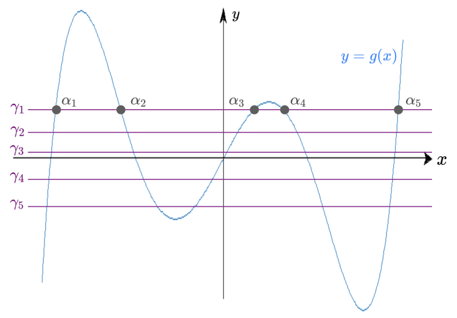

Fig. 2: In Example 1, we have to find a degree-5 polynomial and $\gamma_1,\cdots,\gamma_5$ which satisfies $g(\mathcal{A}_i)=\gamma_i$. The plot shows one possible choice of $g(x)$ and $\gamma_1,\cdots,\gamma_5$. After choosing $g(x)$ and γ_i's ($i=1,\cdots,5$), α_j's are automatically decided ($j=1,\cdots,25$). For instance, $\mathcal{A}_1=\{\alpha_1,\cdots,\alpha_5\}$ are shown on the plot.

A. Locally Recoverable Polynomial Codes

We first give an example of locally recoverable Polynomial codes for $m=4$ and $r=4$ with $P=25$ worker nodes.

Example 1. ($m=4,r=4,P=25$)
We first split the matrices $\mathbf{A}$ and $\mathbf{B}$ into 4 blocks as follows:

$$\mathbf{A}=\begin{bmatrix}\mathbf{A}_1\\\mathbf{A}_2\\\mathbf{A}_3\\\mathbf{A}_4\end{bmatrix},\quad \mathbf{B}=[\mathbf{B}_1\ \mathbf{B}_2\ \mathbf{B}_3\ \mathbf{B}_4], \tag{9}$$

where $\mathbf{A}_i$'s and $\mathbf{B}_i$'s are $N/4\times N$ and $N\times N/4$ dimensional submatrices, respectively. Let $\mathcal{A}=\{\alpha_1,\cdots,\alpha_{25}\}$ be a set of 25 distinct real numbers and let $\mathcal{A}_1=\{\alpha_1,\cdots,\alpha_5\},\cdots,\mathcal{A}_5=\{\alpha_{21},\cdots,\alpha_{25}\}$ be subsets of $\mathcal{A}$ that form a partition of $\mathcal{A}$.

Then, we encode the matrices $\mathbf{A}$ and $\mathbf{B}$ with the following polynomials:

$$\begin{aligned}p_{\mathbf{A}}(x)&=\mathbf{A}_1+\mathbf{A}_2x+\mathbf{A}_3x^2+\mathbf{A}_4x^3\\p_{\mathbf{B}}(x)&=\mathbf{B}_1+\mathbf{B}_2g(x)+\mathbf{B}_3g(x)^2+\mathbf{B}_4g(x)^3\end{aligned}$$

where $g(x)$ is a polynomial of degree 5 that satisfies $g(\mathcal{A}_i)=\gamma_i$. An example choice of $g(x)$ and γ_i's is shown in Fig 2.

The i-th worker gets the encoded matrices, which are the evaluations of the polynomials $p_{\mathbf{A}}(x)$ and $p_{\mathbf{B}}(x)$ at $x=\alpha_i$. The i-th worker then computes the following product:

$$p_{\mathbf{C}}(x)=\sum_{i=1}^{4}\sum_{j=1}^{4}\mathbf{A}_i\mathbf{B}_jx^{i-1}g(x)^{j-1} \tag{10}$$

at $x=\alpha_i$ and returns the result to the master node.

The degree of polynomial $p_{\mathbf{C}}(x)$ is $3\cdot5+3=18$, so the master node can recover the coefficients of $p_{\mathbf{C}}(x)$ from its evaluation at any 19 distinct points. Hence, the recovery threshold $K=19$.

To see that locality $r=4$, let us assume that node 3 is erased, and notice that $g(\cdot)$ satisfies $g(\alpha_1)=g(\alpha_2)=g(\alpha_3)=g(\alpha_4)=g(\alpha_5)=\gamma_1$. Now, $p_{\mathbf{C}}(x)$ at $\alpha_1,\cdots,\alpha_5$ can be rewritten as:

$$p_{\mathbf{C}}(x)=\sum_{i=1}^{4}\left(\sum_{j=1}^{4}\mathbf{B}_j\gamma_1^{j-1}\right)\mathbf{A}_ix^{i-1}. \tag{11}$$

Notice that this is a polynomial of degree 3 which can be recovered from evaluation at any four distinct points, and in this case, $\alpha_1, \alpha_2, \alpha_4, \alpha_5$. □

We now provide a general construction of LRC Polynomial codes. Note that our construction is limited to the case when $r = m$.

Construction 4. ***LRC Polynomial code with*** $r = m$
Splitting of the matrices $\mathbf{A}$ and $\mathbf{B}$ follows Construction 1:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}, \quad \mathbf{B} = [\mathbf{B}_1 \ \mathbf{B}_2 \ \dots \ \mathbf{B}_m]. \qquad (12)$$

Let $\mathcal{A} = \{\alpha_1, \cdots \alpha_P\}$ be a set of P distinct real numbers and let $\{\mathcal{A}_1, \cdots, \mathcal{A}_{\frac{P}{r+1}}\}$ be subsets of $\mathcal{A}$ with size $(r+1)$ which form a partition of $\mathcal{A}$. For simplicity, we assume that $(r+1)$ divides P.

We encode matrix $\mathbf{A}$ and $\mathbf{B}$ using the following polynomials:

$$p_{\mathbf{A}}(x) = \sum_{i=1}^{m} \mathbf{A}_i x^{i-1}, \quad p_{\mathbf{B}}(x) = \sum_{i=1}^{m} \mathbf{B}_i g(x)^{i-1}, \qquad (13)$$

where $g(x)$ is a polynomial of degree $(r+1)$ which is constant on each set $\mathcal{A}_i$. The i-th worker gets the evaluation of $p_{\mathbf{A}}(x)$ and $p_{\mathbf{B}}(x)$ at $x = \alpha_i$. Then a worker node computes the following product:

$$p_{\mathbf{C}}(x) = \sum_{i=1}^{m} \sum_{j=1}^{m} \mathbf{A}_i \mathbf{B}_j x^{i-1} g(x)^{j-1}, \qquad (14)$$

and return the result to a master node. □

Before proving the recovery threshold and locality property of LRC Polynomial code, we want to make an important remark on choosing $g(x)$ and $\mathcal{A}$.

Remark 1 (Finding a good polynomial $g(x)$ and a set $\mathcal{A}$). In [15], a major challenge was to find a suitable polynomial g over $\mathbb{F}_q$ while keeping q small. However, in this work we consider real numbers. In $\mathbb{R}$, as long as $g(x) = 0$ has $r+1$ distinct real roots (sufficient but not necessary condition), we can always find $\gamma_1, \cdots, \gamma_{\frac{P}{r+1}}$ and $\mathcal{A}_1, \cdots, \mathcal{A}_{\frac{P}{r+1}}$ that satisfies $g(\mathcal{A}_i) = \gamma_i$ $(i = 1, \cdots \frac{P}{r+1})$.

However, choosing evaluation points (α_i's) that satisfy the above condition can create numerical stability issues. The numerical stability issue is a persistent problem in coded computing when trying to extend the coding technique from finite field to $\mathbb{R}$ [19], [20]. This is because decoding MDS codes close to capacity often leads to matrices that have poor condition number [21]. Adding locality could worsen the problem. As the degree of g becomes large (*i.e.,* large r), the slope of the polynomial $g(\cdot)$ becomes steep very quickly. This will force us to choose α_i's that are very close to each other which can make the resulting Vandermonde matrix close to singular. How much locality effect the stability issue and how to choose a good polynomial $g(\cdot)$ and γ_i's to make the decoding as numerically stable as possible needs to be studied further. □

The following theorem shows the recovery threshold and locality property of the proposed LRC Polynomial code construction.

Theorem 2. *LRC Polynomial code given in Construction 4 achieves locality $r = m$ and recovery threshold $K = m^2 + m - 1$. Hence, this is an optimal LRC code for locality $r = m$.*

Proof. The degree of the polynomial $p_{\mathbf{C}}$ is $(m-1) + (m+1)(m-1) = m^2 + m - 2$. Hence, we can obtain the coefficients of $p_{\mathbf{C}}$ from evaluation at any m^2+m-1 distinct points. Because $x^{i-1}g(x)^{j-1}$ all have distinct degrees, we can decode $\mathbf{A}_i\mathbf{B}_j$ sequentially from the coefficients. First, recover $\mathbf{A}_m\mathbf{B}_m$ from the coefficient of x^{m^2+m-2}, then recover $\mathbf{A}_m\mathbf{B}_{m-1}$ from the coefficient of x^{m^2+m-2} and $\mathbf{A}_m\mathbf{B}_m$ that was already decoded, and so on. Thus, the recovery threshold $K = m^2 + m - 1$.

Locality $r = m$ is guaranteed as (14) follows the form of (8) in Construction 3 by setting $r = m$ and $k/r = m$. The overhead of having locality $r = m$ is $m - 1$ which is $m^2/m - 1 = k/r - 1$. This shows the optimality of the LRC Polynomial code. □

B. Locally Recoverable MatDot Codes

Before giving a general construction of locally recoverable MatDot codes, we want to give a simple example of LRC MatDot codes for $m = 6$ and $r = 3$.

Example 2. (LRC MatDot with $m = 6, r = 3$)
First, we split the matrices $\mathbf{A}$ and $\mathbf{B}$ as follows:

$$\mathbf{A} = [\mathbf{A}_1 \ \mathbf{A}_1 \ \dots \ \mathbf{A}_6], \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_6 \end{bmatrix}.$$

Let $\mathcal{A} = \{\alpha_1, \cdots, \alpha_P\}$ be a set of P distinct real numbers and let $\mathcal{A}_1 = \{\alpha_1, \cdots, \alpha_4\}, \cdots, \mathcal{A}_{\frac{P}{4}} = \{\alpha_{P-3}, \cdots, \alpha_P\}$ be subsets of $\mathcal{A}$ of size 4 that form a partition of $\mathcal{A}$. Let $g(x)$ be a polynomial with degree 4 that is constant on each subset $\mathcal{A}_i$. We encode matrix $\mathbf{A}$ and $\mathbf{B}$ as follows:

$$p_{\mathbf{A}}(x) = (\mathbf{A}_1 + \mathbf{A}_2 x) + (\mathbf{A}_3 + \mathbf{A}_4 x)g(x) + (\mathbf{A}_5 + \mathbf{A}_6 x)g(x)^2,$$
$$p_{\mathbf{B}}(x) = (\mathbf{B}_6 + \mathbf{B}_5 x) + (\mathbf{B}_4 + \mathbf{B}_3 x)g(x) + (\mathbf{B}_2 + \mathbf{B}_1 x)g(x)^2.$$

The i-th worker node receives the encoded matrices, $p_{\mathbf{A}}(\alpha_i)$ and $p_{\mathbf{B}}(\alpha_i)$, and then computes the following product:

$$\begin{aligned} p_{\mathbf{C}}(x) =& (\mathbf{A}_1 + \mathbf{A}_2 x)(\mathbf{B}_6 + \mathbf{B}_5 x) + \cdots \\ &+ (\mathbf{A}_1\mathbf{B}_1 + \cdots \mathbf{A}_6\mathbf{B}_6) x g(x)^2 + \cdots \\ &+ (\mathbf{A}_5 + \mathbf{A}_6 x)(\mathbf{B}_2 + \mathbf{B}_1 x) g(x)^4 \end{aligned} \qquad (15)$$

at $x = \alpha_i$. Notice that the coefficient of $xg(x)^2$ in (15) is $\mathbf{C} = \mathbf{A}_1\mathbf{B}_1 + \cdots \mathbf{A}_6\mathbf{B}_6$. The degree of polynomial $p_{\mathbf{C}}$ is $4 \cdot 4 + 2 = 18$, so we can recover the coefficients of the polynomial with evaluation at any 19 distinct points. After obtaining the coefficients of $p_{\mathbf{C}}$, the coefficients of $x^i g(x)^j$

for $i = 0, 1, 2, j = 0, \cdots, 4$ can be obtained as they all have distinct degrees. Hence, the recovery threshold $K = 19$.

To see the locality property, let us assume that node 3 is erased, and let us denote $g(\mathcal{A}_1) = \gamma$, *i.e.,* $g(\alpha_1) = g(\alpha_2) = g(\alpha_3) = g(\alpha_4) = \gamma$. Then $p_{\mathbf{C}}(x)$ at $\alpha_1, \cdots, \alpha_4$ can be rewritten as:

$$\begin{aligned} p_{\mathbf{C}}(x) =& (\mathbf{A}_1 + \mathbf{A}_2 x)(\mathbf{B}_6 + \mathbf{B}_5 x) + \cdots \\ &+ (\mathbf{A}_1\mathbf{B}_1 + \cdots \mathbf{A}_6\mathbf{B}_6) x\gamma^2 + \cdots \\ &+ (\mathbf{A}_5 + \mathbf{A}_6 x)(\mathbf{B}_2 + \mathbf{B}_1 x)\gamma^4. \end{aligned}$$

Now, notice that this is a polynomial of degree 2, which can be recovered from evaluation at any three points, and in this case, $\alpha_1, \alpha_2, \alpha_4$. □

We now give a construction of LRC MatDot codes with general m and r. Unlike LRC Polynomial codes, in the LRC MatDot code construction, r can take any value between 1 and $2m - 1$.

Construction 5. ***LRC MatDot Codes***
Splitting of the matrices $\mathbf{A}$ and $\mathbf{B}$ follows Construction 2:

$$\mathbf{A} = [\mathbf{A}_1 \ \mathbf{A}_2 \ \ldots \ \mathbf{A}_m], \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_m \end{bmatrix}. \tag{16}$$

Let $\mathcal{A} = \{\alpha_1, \cdots \alpha_P\}$ be a set of P distinct real numbers and let $\mathcal{A}_1, \cdots, \mathcal{A}_{\frac{P}{r+1}}$ be subsets of $\mathcal{A}$ with size $(r+1)$ which form a partition of $\mathcal{A}$. For simplicity, we assume that $r+1$ divides both P and m. Let $g(x)$ be a polynomial of degree $r+1$ which is constant on each subset $\mathcal{A}_i$. The encoding polynomials of the matrices $\mathbf{A}$ and $\mathbf{B}$ are as follows:

$$\begin{aligned} p_{\mathbf{A}}(x) =& (\mathbf{A}_1 + \cdots + \mathbf{A}_{\frac{r+1}{2}} x^{\frac{r-1}{2}}) \\ &+ (\mathbf{A}_{\frac{r+1}{2}+1} + \cdots + \mathbf{A}_{(r+1)} x^{\frac{r-1}{2}}) g(x) + \cdots \\ &+ (\mathbf{A}_{m-\frac{r+1}{2}+1} + \cdots + \mathbf{A}_m x^{(r-1)/2}) g(x)^{\frac{2m}{r+1}-1}, \end{aligned} \tag{17}$$

$$\begin{aligned} p_{\mathbf{B}}(x) =& (\mathbf{B}_m + \cdots + \mathbf{B}_{m-\frac{r+1}{2}+1} x^{\frac{r-1}{2}}) \\ &+ (\mathbf{B}_{m-\frac{r+1}{2}} + \cdots + \mathbf{B}_{m-r} x^{\frac{r-1}{2}}) g(x) + \cdots \\ &+ (\mathbf{B}_{\frac{r+1}{2}} + \cdots + \mathbf{B}_1 x^{\frac{r-1}{2}}) g(x)^{\frac{2m}{r+1}-1}. \end{aligned} \tag{18}$$

The i-th worker gets the evaluation of $p_{\mathbf{A}}(x)$ and $p_{\mathbf{B}}(x)$ at $x = \alpha_i$ $(i = 1, \cdots, P)$. Then a worker node computes the following product:

$$\begin{aligned} p_{\mathbf{C}}(x) &= p_{\mathbf{A}}(x) p_{\mathbf{B}}(x) \\ &= \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_j x^{j-1} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{B}_{m-j+1} x^{j-1} + \cdots \\ &\quad + (\mathbf{A}_1\mathbf{B}_1 + \cdots + \mathbf{A}_m\mathbf{B}_m) x^{\frac{r-1}{2}} g(x)^{\frac{2m}{r+1}-1} + \cdots \\ &\quad + \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_{m-j+1} x^{j-1} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{B}_j x^{j-1} g(x)^{\frac{4m}{r+1}-2}, \end{aligned} \tag{19}$$

and return the result to a master node. □

The following theorem shows the locality and recovery threshold of the proposed LRC MatDot code construction.

Theorem 3. *The LRC MatDot code given in Construction 5 achieves locality r and recovery threshold $K = 4m - r - 2$.*

Proof. The degree of the polynomial $p_{\mathbf{C}}$ in (19) is $(r-1) + (\frac{4m}{r+1} - 2)(r+1) = 4m - r - 3$, so with evaluation at any $4m - r - 2$ distinct points, the coefficients of $p_{\mathbf{C}}$ can be recovered. Also, notice that the coefficient of $x^{\frac{r-1}{2}} g(x)^{\frac{2m}{r+1}-1}$ is $\mathbf{C} = \mathbf{A}_1\mathbf{B}_1 + \cdots + \mathbf{A}_m\mathbf{B}_m$. As $x^i g(x)^j$ all have distinct degrees for $i = 0, \cdots \frac{r-1}{2}, j = 0, \cdots, \frac{4m}{r+1} - 2$, we can obtain the coefficient of $x^i g(x)^j$ from the coefficients of $p_{\mathbf{C}}(x)$. Hence, the recovery threshold is $K = 4m - r - 2$.

Now, let us show that the locality of the construction is r. The polynomial $p_{\mathbf{C}}(x)$ can be rewritten as:

$$\begin{aligned} p_{\mathbf{C}}(x) =& \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_j x^{j-1} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{B}_{m-j+1} x^{j-1} \\ &+ \Big(\sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_j x^{j-1} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{B}_{m-\frac{r+1}{2}-j+1} x^{j-1} \\ &\quad + \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_{\frac{r+1}{2}+j} x^{j-1} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{B}_{m-j+1} x^{j-1} \Big) g(x) \\ &+ \cdots + \Big(\sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_{m-j+1} x^{j-1} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{B}_j x^{j-1} \Big) g(x)^{\frac{4m}{r+1}-2} \\ =& f_1(x) + f_2(x) g(x) + \cdots + f_{\frac{4m}{r+1}-1}(x) g(x)^{\frac{4m}{r+1}-2}. \end{aligned}$$

Notice that $f_1, \cdots, f_{\frac{4m}{r+1}-1}$ are all polynomials of degree $r-1$. Let $p_{\mathbf{C}}(\alpha)$ be the lost matrix and let $\alpha \in \mathcal{A}_l$. For all $\beta \in \mathcal{A}_l$,

$$p_{\mathbf{C}}(\beta) = f_1(\beta) + f_2(\beta)\gamma_l + \cdots + f_{\frac{4m}{r+1}-1}(\beta)\gamma_l^{\frac{4m}{r+1}-2}, \tag{20}$$

because $g(\mathcal{A}_l) = \gamma_l$. This is a degree-$(r-1)$ polynomial in β, so the coefficients of $p_{\mathbf{C}}$ can be recovered from its evaluation at the r points in $\mathcal{A}_l \backslash \{\alpha\}$. Then the lost matrix can be recovered by evaluating $p_{\mathbf{C}}(\beta)$ given in (20) at $\beta = \alpha$. □

By comparing the recovery threshold given in Theorem 3 and the recovery threshold of MatDot codes without locality (Construction 2), we can see that the overhead of having locality r is $2m - r - 1$. However, the optimal overhead suggested by Theorem 1 is $\lceil \frac{2m-1}{r} \rceil - 1$. Thus, there is a gap between the proposed LRC MatDot codes and the optimal LRC codes; while the optimal overhead of having locality r decreases in the order of $1/r$, the overhead of LRC MatDot codes decreases linearly in r (see Fig 3). Whether this sub-optimality is inevitable due to the structure of MatDot codes is an open question.

C. Locally Recoverable Systematic MatDot Codes

In this section, we consider the systematic encoding of LRC MatDot codes. Although we assume a master-worker system in this paper, LRC matrix multiplication nodes will

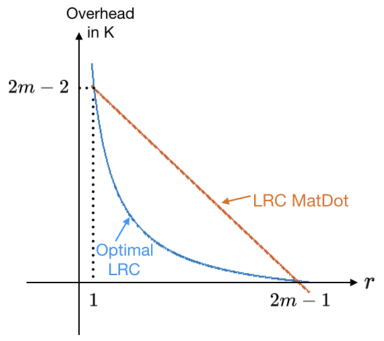

Fig. 3: This plot shows the gap between the optimal LRC codes and the proposed LRC MatDot construction. In the optimal LRC codes, the overhead of having locality r in K is $\lceil \frac{2m-1}{r} \rceil - 1$, while in the LRC MatDot codes, the overhead is $2m-1-r$.

be also valuable in a fully-distributed system that does not have a master node. Systematic encoding would be particularly useful in this setting because we can obtain the final computation output by only repairing failed systematic nodes. Assuming that failure rate is low, locality will let us repair a failed systematic node by communicating with only a few other nodes instead of communicating with all the other nodes.

We will follow the definition of systematic MatDot codes given in [2]. For the matrix multiplication problem stated in Section II-A and when the matrices $\mathbf{A}$ and $\mathbf{B}$ are split as in (4), we say a code $\mathcal{C}$ is systematic when there are m systematic nodes whose outputs are $\mathbf{A}_i\mathbf{B}_i$ for $i = 1, \cdots, m$. Namely, there is a set $\mathcal{A}_{\text{sys}} = \{\beta_1, \cdots, \beta_m\} \subseteq \mathcal{A}$ such that $p_{\mathbf{C}}(\beta_i) = \mathbf{A}_i\mathbf{B}_i$. In [2], systematic MatDot codes were constructed by exploiting Lagrange's interpolation. For more details, we refer to [2, Section IV].

We will first give an example of systematic LRC MatDot codes for $m = 4, r = 3$ with $P = 16$ worker nodes.

Example 3. (Systematic LRC MatDot Codes with $m = 4, r = 3, P = 16$)
We first split matrices $\mathbf{A}$ and $\mathbf{B}$ into 4 blocks as follows:

$$\mathbf{A} = [\mathbf{A}_1 \ \mathbf{A}_1 \ \mathbf{A}_3 \ \mathbf{A}_4], \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \\ \mathbf{B}_4 \end{bmatrix} \tag{21}$$

where $\mathbf{A}_i$'s and $\mathbf{B}_i$'s are $N \times N/4$ and $N/4 \times N$ dimensional submatrices, respectively. Let $\mathcal{A} = \{\alpha_1, \cdots, \alpha_{16}\}$ be a set of 16 distinct real numbers and let $\mathcal{A}_1 = \{\alpha_1, \cdots, \alpha_4\}, \cdots, \mathcal{A}_4 = \{\alpha_{13}, \cdots, \alpha_{16}\}$ be disjoint subsets of $\mathcal{A}$ that form a partition of $\mathcal{A}$. Let $g(x)$ be a polynomial of degree 4 which satisfies: $g(\mathcal{A}_i) = \gamma_i$ for $i = 1, \cdots, 4$. Then, we encode the matrices $\mathbf{A}$ and $\mathbf{B}$ with the following polynomials:

$$\begin{aligned} p_{\mathbf{A}}(x) &= \big(\mathbf{A}_1 \frac{x-\alpha_1}{\alpha_2-\alpha_1} + \mathbf{A}_2 \frac{x-\alpha_1}{\alpha_2-\alpha_1}\big) f_1(x) \\ &+ \big(\mathbf{A}_3 \frac{x-\alpha_6}{\alpha_5-\alpha_6} + \mathbf{A}_4 \frac{x-\alpha_5}{\alpha_6-\alpha_5}\big) f_2(x) \end{aligned}$$

$$\begin{aligned} p_{\mathbf{B}}(x) &= \big(\mathbf{B}_1 \frac{x-\alpha_1}{\alpha_2-\alpha_1} + \mathbf{B}_2 \frac{x-\alpha_2}{\alpha_1-\alpha_2}\big) f_1(x) \\ &+ \big(\mathbf{B}_3 \frac{x-\alpha_6}{\alpha_5-\alpha_6} + \mathbf{B}_4 \frac{x-\alpha_5}{\alpha_6-\alpha_5}\big) f_2(x) \end{aligned}$$

where $f_1(x) = \lambda_{11} + \lambda_{12} g(x)$ and $f_2(x) = \lambda_{21} + \lambda_{22} g(x)$. The coefficients λ_{ij}'s are chosen so that $f_i(\mathcal{A}_j) = \delta_{ij}$ for $i, j = 1, 2$. They can be obtained by solving:

$$\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ \gamma_1 & \gamma_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The i-th worker node receives the encoded matrices, $p_{\mathbf{A}}(\alpha_i)$ and $p_{\mathbf{B}}(\alpha_i)$, and then computes the following product:

$$\begin{aligned} p_{\mathbf{c}}(x) &= p_{\mathbf{A}}(x) p_{\mathbf{B}}(x) \\ &= \big(\mathbf{A}_1 \frac{x-\alpha_1}{\alpha_2-\alpha_1} + \mathbf{A}_2 \frac{x-\alpha_1}{\alpha_2-\alpha_1}\big)\big(\mathbf{B}_1 \frac{x-\alpha_1}{\alpha_2-\alpha_1} \\ &+ \mathbf{B}_2 \frac{x-\alpha_2}{\alpha_1-\alpha_2}\big) f_1(x)^2 + \big(\ \cdots \ \big) f_1(x) f_2(x) \\ &+ \big(\mathbf{A}_3 \frac{x-\alpha_6}{\alpha_5-\alpha_6} + \mathbf{A}_4 \frac{x-\alpha_5}{\alpha_6-\alpha_5}\big)\big(\mathbf{B}_3 \frac{x-\alpha_6}{\alpha_5-\alpha_6} \\ &+ \mathbf{B}_4 \frac{x-\alpha_5}{\alpha_6-\alpha_5}\big) f_2(x)^2 \end{aligned}$$

at $x = \alpha_i$. First, note that the following holds:

$$\begin{aligned} p_{\mathbf{c}}(\alpha_1) = \mathbf{A}_1\mathbf{B}_1, \ \ p_{\mathbf{c}}(\alpha_2) = \mathbf{A}_2\mathbf{B}_2, \\ p_{\mathbf{c}}(\alpha_5) = \mathbf{A}_3\mathbf{B}_3, \ \ p_{\mathbf{c}}(\alpha_6) = \mathbf{A}_4\mathbf{B}_4. \end{aligned}$$

Hence, this is a systematic code. The degree of $p_{\mathbf{C}}(\cdot)$ is $2 \cdot 4 + 2 = 10$, so with evaluation at any 11 points, we can recover the coefficients on $p_{\mathbf{C}}(x)$. The recovery threshold $K = 11$.

Now, to examine the locality property, let as assume that node 3 is erased. Because f_1 and f_2 are linear combinations of constant and $g(x)$, they are also constant on each subset $\mathcal{A}_i$. For $i = 1, \cdots, 4$, $p_{\mathbf{C}}(\alpha_i)$ becomes a polynomial of degree 2 in α_i as $f_1(\alpha_i), f_2(\alpha_i)$ are constant for $\alpha_1, \cdots, \alpha_4$. Hence, the coefficients of $p_{\mathbf{C}}$ can be recovered from three evaluations, $p_{\mathbf{C}}(\alpha_1), p_{\mathbf{C}}(\alpha_2)$, and $p_{\mathbf{C}}(\alpha_4)$, and thus the lost matrix $p_{\mathbf{C}}(\alpha_3)$ can be recovered. □

We now give a construction of systematic LRC MatDot codes for general m and r.

Construction 6. ***Systematic LRC MatDot Codes***
The key idea is to replace $g(x)^{i-1}$ in $p_{\mathbf{A}}(x)$ given in (17) with $f_i(x)$ which satisfies $f_i(\mathcal{A}_j) = \delta_{ij}$ $(i, j = 1, \cdots, \frac{2m}{r+1})$, and to replace $\sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_{\frac{r+1}{2}(i-1)+j} x^{j-1}$ in (17) with Lagrange interpolation $(i = 1, \cdots, \frac{2m}{r+1})$. We do similar replacements for $p_{\mathbf{B}}(x)$ in (18). Let us explain this in more detail.

First, we generate f_i's by linearly combining $1, g(x), \cdots, g(x)^{\frac{2m}{r+1}-1}$: $f_i(x) = \sum_{j=1}^{\frac{2m}{r+1}} \lambda_{ij} g(x)^{j-1}$. Let us denote $g(\mathcal{A}_i) = \gamma_i$. We obtain λ_{ij}'s by solving the following equation:

$$\Lambda \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \gamma_1 & \gamma_2 & \cdots & \gamma_{\frac{2m}{r+1}} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_1^{\frac{2m}{r+1}-1} & \gamma_2^{\frac{2m}{r+1}-1} & \cdots & \gamma_{\frac{2m}{r+1}}^{\frac{2m}{r+1}-1} \end{bmatrix} = I_{\frac{2m}{r+1}},$$

where $\Lambda = \left[\lambda_{ij}\right]$ and $I_{\frac{2m}{r+1}}$ is an identity matrix of dimension $\frac{2m}{r+1} \times \frac{2m}{r+1}$. It is easy to see that by this choice of λ_{ij}'s, $f_i(\mathcal{A}_j) = \delta_{ij}$ for $i, j = 1, \cdots, \frac{2m}{r+1}$.

Let $\widetilde{\mathcal{A}}_i$ be a subset of $\mathcal{A}_i$ which has the first half of the elements, that is, $\widetilde{\mathcal{A}}_i = \{\alpha_{(r+1)(i-1)+1}, \cdots, \alpha_{(r+1)(i-1)+\frac{r+1}{2}}\}$, and let $\widetilde{\mathcal{A}}_i(j) = \widetilde{\mathcal{A}}_i \backslash \{\alpha_{(r+1)(i-1)+j}\}$. Now, let us define $\phi_{ij}(x)$ as follows:

$$\phi_{i,j}(x) = \prod_{\alpha \in \widetilde{\mathcal{A}}_i(j)} \frac{x-\alpha}{\alpha_{(r+1)(i-1)+j} - \alpha}.$$

Then, we encode $\mathbf{A}$ and $\mathbf{B}$ using the following polynomials:

$$\begin{aligned} p_{\mathbf{A}}(x) &= \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_j \phi_{1,j}(x) f_1(x) + \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_{\frac{r+1}{2}+j} \phi_{2,j}(x) f_2(x) \\ &\quad + \cdots + \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_{m-\frac{r+1}{2}+j} \phi_{\frac{2m}{r+1},j}(x) f_{\frac{2m}{r+1}}(x) \\ &= \sum_{i=1}^{\frac{2m}{r+1}} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_{\frac{r+1}{2}(i-1)+j} \phi_{i,j}(x) f_i(x), \quad (22) \\ p_{\mathbf{B}}(x) &= \sum_{i=1}^{\frac{2m}{r+1}} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{B}_{\frac{r+1}{2}(i-1)+j} \phi_{i,j}(x) f_i(x). \quad (23) \end{aligned}$$

The i-th worker receives the evaluation of $p_{\mathbf{A}}(x)$ and $p_{\mathbf{B}}(x)$ at $x = \alpha_i$. A worker node then computes the following product:

$$\begin{aligned} p_{\mathbf{C}}(x) &= p_{\mathbf{A}}(x) p_{\mathbf{B}}(x) \\ &= \Big(\sum_{i=1}^{\frac{2m}{r+1}} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{A}_{\frac{r+1}{2}(i-1)+j} \phi_{i,j}(x) f_i(x) \Big) \\ &\quad \Big(\sum_{i=1}^{\frac{2m}{r+1}} \sum_{j=1}^{\frac{r+1}{2}} \mathbf{B}_{\frac{r+1}{2}(i-1)+j} \phi_{i,j}(x) f_i(x) \Big), \end{aligned}$$

and returns the result to the master node. $\square$

The following theorem shows that the Construction 6 is indeed systematic, and achieves the same locality and recovery threshold as the non-systematic version LRC MatDot codes.

Theorem 4. *The systematic LRC MatDot code given in Construction 6 is systematic, and achieves locality r and recovery threshold $K = 4m - r - 2$.*

Proof. To show that the construction is systematic, we have to show that there exists a subset $\mathcal{A}_{\text{sys}} = \{\beta_1, \cdots, \beta_m\} \subseteq \mathcal{A}$ such that $p_{\mathbf{C}}(\beta_i) = \mathbf{A}_i \mathbf{B}_i$. Let $\mathcal{A}_{\text{sys}} = \widetilde{\mathcal{A}}_1 \bigcup \cdots \bigcup \widetilde{\mathcal{A}}_{\frac{2m}{r+1}}$. Now, notice that for $i = 1, \cdots, \frac{2m}{r+1}$ and $j = 1, \cdots, \frac{r+1}{2}$,

$$\begin{aligned} p_{\mathbf{C}}(\alpha_{(r+1)(i-1)+j}) &= p_{\mathbf{A}}(\alpha_{(r+1)(i-1)+j}) p_{\mathbf{B}}(\alpha_{(r+1)(i-1)+j}) \\ &= \mathbf{A}_{\frac{r+1}{2}(i-1)+j} \mathbf{B}_{\frac{r+1}{2}(i-1)+j}, \end{aligned}$$

and $\alpha_{(r+1)(i-1)+j} \in \mathcal{A}_{\text{sys}}$. This proves that the code is systematic.

The degree of $\phi_{i,j}$'s is $\frac{r-1}{2}$ and the degree of f_i's is $(r+1)(\frac{2m}{r+1} - 1) = 2m - r - 1$. Thus, the degree of $p_{\mathbf{C}}(x)$ is $2 \cdot (\frac{r-1}{2} + 2m - r - 1) = 4m - r - 3$. This shows that the recovery threshold $K = 4m - r - 2$.

Finally, let us show the locality property. Let $p_{\mathbf{C}}(\alpha)$ be the lost symbol and let $\alpha \in \mathcal{A}_l$. Then, for all $\beta \in \mathcal{A}_l$, $f_i(\beta) = \sum_{j=1}^{\frac{2m}{r+1}} \lambda_{ij} \gamma_l^{j-1} = \psi_i$. Then, $p_{\mathbf{C}}(\beta)$ can be rewritten as:

$$\begin{aligned} p_{\mathbf{C}}(\beta) = \Big(\sum_{j=1}^{\frac{r+1}{2}} \big(\sum_{i=1}^{\frac{2m}{r+1}} \psi_i \mathbf{A}_{\frac{r+1}{2}(i-1)+j} \big) \phi_{i,j}(\beta) \Big) \cdot \\ \Big(\sum_{j=1}^{\frac{r+1}{2}} \big(\sum_{i=1}^{\frac{2m}{r+1}} \psi_i \mathbf{B}_{\frac{r+1}{2}(i-1)+j} \big) \phi_{i,j}(\beta) \Big). \end{aligned}$$

Since $\phi_{i,j}$'s are polynomials of degree $\frac{r-1}{2}$, $p_{\mathbf{C}}(\beta)$ is a degree-$(r-1)$ polynomial in β. Hence, from the evaluation of $p_{\mathbf{C}}(\cdot)$ at the r points in $\mathcal{A}_l \backslash \{\alpha\}$, we can recover the coefficients of $p_{\mathbf{C}}(\cdot)$. The lost symbol $p_{\mathbf{C}}(\alpha)$ can then be recovered by evaluating $p_{\mathbf{C}}(\beta)$ at $\beta = \alpha$. $\square$

IV. Conclusion and Future Work

In this work, we proposed LRC Polynomial codes and LRC MatDot codes. The proposed coding strategies have a few limitations. First, LRC Polynomial codes are only limited to the case when $r = m$. Designing LRC Polynomial codes with different r values can be studied in the future. Also, we do not have the systematic version of LRC Polynomial codes, which we believe to have the most practical usage. For LRC MatDot codes, we can choose different r values ranging from 1 to $2m-1$, but the recovery threshold K has a gap from optimal LRC codes. Finding an improved strategy or proving that this is optimal for MatDot codes is an interesting open question.

As mentioned in Remark 1, in order to put the proposed coding techniques into practice, numerical stability of the constructions should be understood. While this issue is present in any polynomial-based real-number codes, and not just limited to LRC codes, understanding the effect of locality on the numerical stability poses a different question. Lastly, extending the proposed techniques to LRC codes with higher availability is an intriguing future direction.

References

[1] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," in Communication, Control, and Computing (Allerton), 2017 55th Annual Allerton Conference on. IEEE, 2017, pp. 1264–1270.

[2] 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.

[3] 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.

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

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

[6] 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 IEEE International Symposium on Information Theory (ISIT), 2018.

[7] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes," 2018.

[8] 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.

[9] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in INFOCOM, 2011 Proceedings IEEE. IEEE, 2011, pp. 1215–1223.

[10] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," IEEE Transactions on Information Theory, vol. 58, no. 11, pp. 6925–6934, 2012.

[11] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," ACM Transactions on Storage (TOS), vol. 9, no. 1, p. 3, 2013.

[12] N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, "Optimal locally repairable codes via rank-metric codes," in Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. IEEE, 2013, pp. 1819–1823.

[13] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," IEEE Transactions on Information Theory, vol. 60, no. 10, pp. 5843–5855, Oct 2014.

[14] P. Gopalan, C. Huang, B. Jenkins, and S. Yekhanin, "Explicit maximally recoverable codes with locality." IEEE Trans. Information Theory, vol. 60, no. 9, pp. 5245–5256, 2014.

[15] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," IEEE Transactions on Information Theory, vol. 60, no. 8, pp. 4661–4676, 2014.

[16] V. R. Cadambe and A. Mazumdar, "Bounds on the size of locally recoverable codes," IEEE transactions on information theory, vol. 61, no. 11, pp. 5787–5794, 2015.

[17] I. Tamo, A. Barg, S. Goparaju, and R. Calderbank, "Cyclic lrc codes and their subfield subcodes," in Information Theory (ISIT), 2015 IEEE International Symposium on. IEEE, 2015, pp. 1262–1266.

[18] 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.

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

[20] M. Haikin and R. Zamir, "Analog coding of a source with erasures," in 2016 IEEE International Symposium on Information Theory (ISIT). IEEE, 2016, pp. 2074–2078.

[21] W. Gautschi, "How (un) stable are vandermonde systems," Asymptotic and computational analysis, vol. 124, pp. 193–210, 1990.