Numerically Stable Polynomially Coded Computing

Mohammad Fahim, *Student Member, IEEE*, and Viveck R. Cadambe, *Member, IEEE*

***Abstract*— We study the numerical stability of polynomial based encoding methods, which has emerged to be a powerful class of techniques for providing straggler and fault tolerance in the area of coded computing. Our contributions are as follows:**

1) **We construct new codes for matrix multiplication that achieve the same fault/straggler tolerance as the previously constructed *MatDot Codes* and *Polynomial Codes*.**
2) **We show that the condition number of every $m \times m$ sub-matrix of an $m \times n, n \geq m$ Chebyshev-Vandermonde matrix, evaluated on the n-point Chebyshev grid, grows as $O(n^{2(n-m)})$ for $n > m$.**
3) **By specializing our orthogonal polynomial based constructions to Chebyshev polynomials, and using our condition number bound for Chebyshev-Vandermonde matrices, we construct new numerically stable techniques for coded matrix multiplication. We empirically demonstrate that our constructions have significantly lower numerical errors compared to previous approaches which involve inversion of Vandermonde matrices. We generalize our constructions to explore the trade-off between computation/communication and fault-tolerance.**
4) **We propose a numerically stable specialization of Lagrange coded computing. Our approach involves the choice of evaluation points and a suitable decoding procedure. Our approach is demonstrated empirically to have lower numerical errors as compared to standard methods.**

***Index Terms*— Distributed computing, coded computing, fault tolerance, stragglers, numerical stability.**

I. Introduction

THE recently emerging area of "coded computing" focuses on incorporating redundancy based on coding-theory-inspired strategies to tackle central challenges in distributed computing, including stragglers, failures, processing errors, communication bottlenecks and security issues. Such ideas have been applied to different large scale distributed computations such as matrix multiplication [1]–[6], gradient methods [7], [8], linear solvers [9]–[11] and multi-variate polynomial evaluation [12]. An important idea that has emerged from this body of the work is the use of novel, Reed-Solomon like *polynomial* based methods for encoding data. In polynomial based methods, each computation node stores a linearly encoded combination of the data partitions, where data stored at different worker nodes can be interpreted as evaluation of an appropriate polynomial at different points. The nodes then perform computation on these encoded versions of the data, and a central master/fusion node aggregates the outputs of these computations to recover the overall result via a decoding process that inevitably involves polynomial interpolation. Much like Reed-Solomon Codes, if the number of nodes performing the computation is higher than the number of evaluation points required for accurate interpolation, the overall computation is tolerant to faults and stragglers.

Perhaps the most striking application of polynomial based methods comes in the context of matrix multiplication. To multiply two $N \times N$ matrices $\mathbf{A}, \mathbf{B}$, assuming that each node stores $1/m$ fraction of each matrix, classical work in algorithm based fault tolerance [13] outlines a coding based method which has been analyzed in [14].

Reference [6] showed through *polynomial* based encoding methods that the result of just m^2 nodes can be used by the master node to recover the matrix-product. Remarkably, this means that polynomial based codes ensure that *the recovery threshold* - the worst case number of nodes whose computation suffices to recover the overall matrix-product - does not grow with P, the number of the distributed system's worker nodes, unlike the approaches of [13], [14]. The recovery threshold for matrix multiplication has been improved to $2m - 1$ via a code construction called MatDot Codes in [3], albeit at a higher communication/computation cost than codes in [6]. A second prominent application of polynomial based methods is the idea of *Lagrange coded computing* [12], where coding is applied for multi-variate polynomial computing with guarantees of straggler resilience, security and privacy. In addition, polynomial-based methods are also useful for communication-efficient approaches for inverse problems and gradient methods [8], [10], [15].

Despite the enormous success, the scalability of polynomial based methods in practice are often limited by an "inconvenient truth", their numerical instability. The decoding methods for polynomial based methods require interpolating a degree $K - 1$ polynomial using K evaluation points. While this is numerically stable for classical error correcting codes for communication and storage which are implemented over finite fields, we are concerned here for data processing applications where the operations are typically real-valued. A common reason for the instability is that either implicitly or explicitly, interpolation often solves a linear system whose transform is characterized by a Vandermonde matrix. It is well known that the condition number of Vandermonde matrices with real-

Manuscript received May 22, 2019; revised September 7, 2020; accepted December 26, 2020. Date of publication January 11, 2021; date of current version April 21, 2021. This work was supported by NSF under Grant CCF 1763657. This article was presented in part at the 2019 IEEE International Symposium on Information Theory (ISIT). *(Corresponding author: Mohammad Fahim.)*

Mohammad Fahim was with the Department of Electrical Engineering, The Pennsylvania State University, University Park, PA 16802 USA. He is now with Qualcomm Technologies, Inc., San Diego, CA 92121 USA (e-mail: mfahim@qti.qualcomm.com).

Viveck R. Cadambe is with the Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802 USA.

Communicated by P. Sadeghi, Associate Editor for Coding Techniques.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2021.3050526.

Digital Object Identifier 10.1109/TIT.2021.3050526

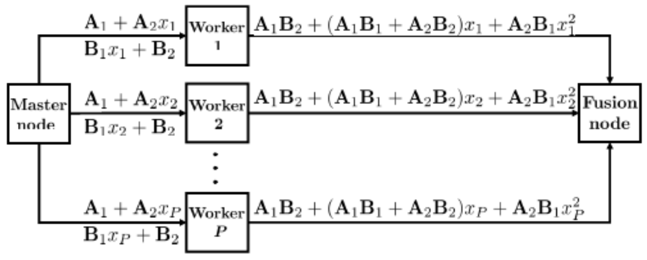
valued nodes grows exponentially in the dimension of the matrix [16]–[19]. The large condition number means that small perturbations of the Vandermonde matrix due to numerical precision errors can result in singular matrices [20], [21]. In practice, this can translate to large numerical errors even when the coded computation is distributed among few tens of nodes[1]. Engineering intuition dictates that the main scalability bottlenecks in distributed computing include computation cost per worker, communication bottlenecks, and stragglers. However, for *polynomially coded computing*, it turns out that numerically stability can constitute a critical bottleneck for scalability of such codes. Indeed, a polynomially coded computing scheme that achieves the minimum recovery threshold, and that is optimal in terms of computation/commununication, may incur large numerical errors when implemented on beyond tens of computing nodes. For example, our experiments show that applying MatDot codes [3] for distributed matrix multiplication on systems with a number of worker nodes larger than 30 yields huge numerical errors[2].

Remark 1.1: In this paper, we are only interested in real-valued code constructions that are implemented on systems that use floating point representation for numbers. Moreover, to the best of our knowledge, this is the first work to study the conditioning of such polynomially coded computing problem.

II. Summary of Contributions

In this paper, we develop a new, numerically stable, approach for polynomially coded computing. Our proposed approach is relevant to several coded computing code constructions that use polynomials expanded in the monomial basis, which result in inherently ill-conditioned Vandermonde-matrices. Using techniques from numerical approximation theory [20], [21], our paper provides recipes that replace the monomial basis with well-conditioned alternatives that lead to code constructions that are provably more stable. Additionally, our contributions also include recipes for provably stable approaches for existing code constructions that use polynomials expanded in the Lagrange basis. We demonstrate our approach through two important applications of polynomially coded computing: matrix multiplication and Lagrange coded computing.

To illustrate our results, consider the coded matrix multiplication problem, where the goal is to multiply two matrices $\mathbf{A}, \mathbf{B}$ over P computation nodes where each node stores $1/m$ fraction of each of the two matrices. A master node encodes $\mathbf{A}, \mathbf{B}$ into P matrices each, and sends these matrices respectively to each worker node. Each worker node multiplies the received encoded matrices, and sends the product back to the fusion node[3], which aims to recover $\mathbf{AB}$ from a subset of the worker nodes. The recovery threshold is defined as a number K such that the computation of any set of K worker nodes suffices to recover the product $\mathbf{AB}$. The MatDot scheme of [3] achieves the best known recovery threshold of $2m-1$. We begin with an example of MatDot Codes for $m=2$.



$\mathbf{A}_1+\mathbf{A}_2x_1$
$\mathbf{B}_1x_1+\mathbf{B}_2$
Worker 1
$\mathbf{A}_1\mathbf{B}_2+(\mathbf{A}_1\mathbf{B}_1+\mathbf{A}_2\mathbf{B}_2)x_1+\mathbf{A}_2\mathbf{B}_1x_1^2$
Master node
$\mathbf{A}_1+\mathbf{A}_2x_2$
$\mathbf{B}_1x_2+\mathbf{B}_2$
Worker 2
$\mathbf{A}_1\mathbf{B}_2+(\mathbf{A}_1\mathbf{B}_1+\mathbf{A}_2\mathbf{B}_2)x_2+\mathbf{A}_2\mathbf{B}_1x_2^2$
Fusion node
$\mathbf{A}_1+\mathbf{A}_2x_P$
$\mathbf{B}_1x_P+\mathbf{B}_2$
Worker P
$\mathbf{A}_1\mathbf{B}_2+(\mathbf{A}_1\mathbf{B}_1+\mathbf{A}_2\mathbf{B}_2)x_P+\mathbf{A}_2\mathbf{B}_1x_P^2$

Fig. 1. Example of MatDot Codes [3], with a recovery threshold of 3. The matrix product $\mathbf{AB}$ is the coefficient of x in $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$, and can be recovered at the fusion node upon receiving the output of any 3 worker nodes and interpolating $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$.

Example 1 (MatDot Codes [3], Recovery Threshold = 3): Consider two $N \times N$ matrices

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}\mathbf{B}_1 \\ \mathbf{B}_2\end{bmatrix},$$

where $\mathbf{A}_1, \mathbf{A}_2$ are $N \times N/2$ matrices and $\mathbf{B}_1, \mathbf{B}_2$ are $N/2 \times N$ matrices. Define $p_{\mathbf{A}}(x) = \mathbf{A}_1 + \mathbf{A}_2 x$ and $p_{\mathbf{B}}(x) = \mathbf{B}_1 x + \mathbf{B}_2$, and let $x_1, \cdots, x_P$ be distinct real values. Notice that $\mathbf{AB} = \mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2$ is the coefficient of x in polynomial $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$. In MatDot Codes, as illustrated in Fig. 1, worker node i computes $p_{\mathbf{A}}(x_i)p_{\mathbf{B}}(x_i),\ i = 1, 2, \ldots P$, so that from any 3 of the P nodes, the polynomial $p(x) = \mathbf{A}_1\mathbf{B}_2 + (\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2)x + \mathbf{A}_2\mathbf{B}_1\ x^2$ can be interpolated. Having interpolated the polynomial, the product $\mathbf{AB}$ is simply the coefficient of x.

A generalization of the above example leads to a recovery threshold of $2m-1$, with a decoding process that involves effectively inverting a $(2m-1) \times (2m-1)$ Vandermonde matrix. It has been shown that the condition number of the $n \times n$ Vandermonde matrix grows exponentially in n with both ℓ_∞ and ℓ_2 norms [16], [17]. The intuition behind the inherent poor conditioning of the monomial basis $\{1, x, x^2, \ldots, x^{2m-1}\}$ is demonstrated in Fig. 2 and Fig. 3.

Motivated by Fig. 3, in this paper, we aim to choose polynomials that are orthonormal. However, it is not immediately clear whether orthonormal polynomials are applicable for matrix multiplications. We demonstrate the applicability of orthonormal codes for matrix multiplication. For the example below, let $q_0(x), q_1(x)$ denote two orthonormal polynomials such that

$$\int_{-1}^{1} q_i(x)q_j(x)dx = \begin{cases} 0 & \text{if } i=j \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

where $q_i(x), i = 0, 1$ has degree i.

Example 2 (OrthoMatDot Codes [This paper], Recovery Threshold = 3): For two $N \times N$ matrices $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}, \mathbf{B} = \begin{bmatrix}\mathbf{B}_1 \\ \mathbf{B}_2\end{bmatrix}$, let $p_{\mathbf{A}}(x) = \mathbf{A}_1 q_0(x) + \mathbf{A}_2 q_1(x)$ and $p_{\mathbf{B}}(x) = \mathbf{B}_1 q_0(x) + \mathbf{B}_2 q_1(x)$. Notice that because of (1), we have

$$\mathbf{AB} = \int_{-1}^{1} p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)dx.$$

[1]For example, reference [22] reports that "In our experiments we observed large floating point errors when inverting high degree Vandermonde matrices for polynomial interpolation".

[2]It is possible to reduce the numerical errors further for MatDot and other similar codes under some circumstances (See Remark 3.2).

[3]The master and fusion nodes are logical entities; in practice, they may be the same node, or may be emulated in a decentralized manner by the computation nodes.

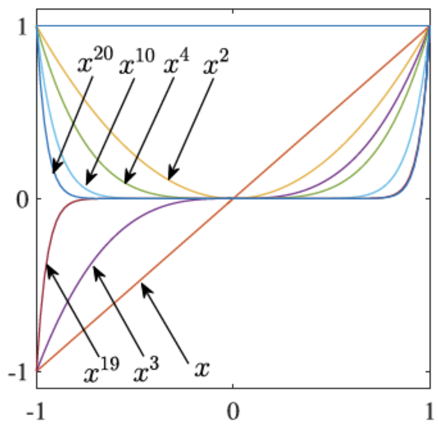


Fig. 2. Plot of monomials $1, x, x^2, x^3, x^4, x^{10}, x^{19}, x^{20}$ versus x for $x \in [-1, 1]$. Note that for a large degree d, small changes in x can lead to large changes in x^d; this leads to significant numerical errors when working with the monomial basis.

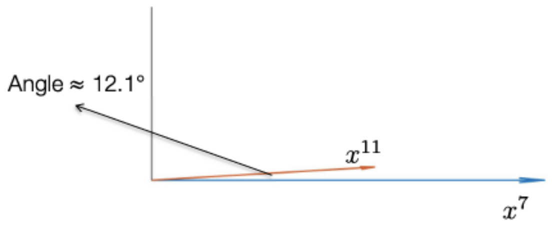


Fig. 3. Note that $\{1, x, \ldots, x^d\}$ forms a basis for the vector space of d-degree polynomials, with the inner-product $\langle f, g\rangle = \int_{-1}^{1} f(x)g(x)dx$. We have plotted the vectors x^7 and x^{11}. The small angle between the two vectors leads to numerical errors.

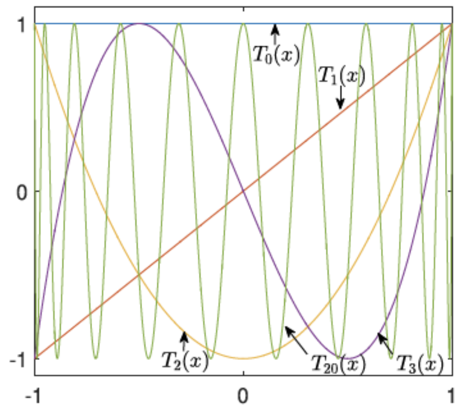


Fig. 4. Plot of Chebyshev polynomials $T_0(x), T_1(x), T_2(x), T_3(x), T_{20}(x)$ versus x for $x \in [-1, 1]$.

This leads to the following coded computing scheme: worker node i computes $p_{\mathbf{A}}(x_i)p_{\mathbf{B}}(x_i)$, $i = 1, 2, \ldots P$, where $x_1, \cdots, x_P$ are distinct real values, so that from any 3 of the P nodes, the fusion node can interpolate $p(x) = p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$. Having interpolated the polynomial, the fusion node obtains the product $\mathbf{AB}$ by performing $\int_{-1}^{1} p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)dx$. This example is illustrated in Fig. 5.

A simple generalization of the above example, described in Construction 1 in Section IV, leads to a class of codes, we refer to it as *OrthoMatDot Codes,* with recovery threshold of $2m - 1$, the same recovery threshold as MatDot Codes. In general, orthonormal polynomials are defined over arbitrary weight measure $\int_{-1}^{1} \cdot\ w(x)dx$; some well known classes of polynomials corresponding to different weight measures $w(x)$ include Legendre, Chebyshev, Jacobi and Laguerre Polynomials [20], [21]. Our OrthoMatDot Codes in Section IV can use any weight measure, and therefore can be used with different classes of orthonormal polynomials. Of particular interest to our paper are the Chebyshev polynomials (Fig. 4) whose numerical properties (discussed in Section III) lead to developing numerically stable code constructions, in this paper, for the polynomially coded computing problem.

With our basic template, the task of developing numerically stable codes boils down to (A) interpolating $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ in a numerically stable manner, and (B) integrating this polynomial in a numerically stable manner. For task (B), we use a decoding procedure via *Gauss Quadrature* [20], [21], [23] to recover the integral. Task (A) is particularly challenging in the coding setting, because our goal is to interpolate the coefficients of $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ - expanded over a series of orthonormal polynomials - from any $2m - 1$ points among a set of P points.

In Section V, we provide a specialization to the class of OrthoMatDot Codes, a numerically stable matrix multiplication code construction that has the same recovery threshold and communication/computation cost per worker as MatDot codes. The construction specializes the class of OrthoMatDot Codes via the use of Chebyshev polynomials, which are a class of orthogonal polynomials that are ubiquitous in numerical methods and approximation theory [21]. Construction 2 also specifies the choice of evaluation points $x_1, x_2, \ldots, x_P$.

The decoding procedure outlined for the specialization of OrthoMatDot Codes in Section V involves the effective inversion of some $(2m - 1) \times (2m - 1)$ sub-matrix of a $(2m - 1) \times P$ *Chebyshev-Vandermonde* matrix [19], where each of the i-th column contains evaluations of the first $2m-1$ Chebyshev polynomials at $x_i, i = 1, 2, \ldots, P$. A key technical result of our paper shows that, with our choice of evaluation points $x_1, x_2, \ldots, x_P$, every $(2m - 1) \times (2m - 1)$ square sub-matrix of the $(2m - 1) \times P$ Chebyshev-Vandermonde matrix is well-conditioned. More precisely, we show that, with our choice of $x_1, x_2, \ldots, x_P$, the condition number of *any* $(2m - 1) \times (2m - 1)$ sub-matrix of the Chebyshev-Vandermonde matrix grows at most polynomially in P when the number of redundant parity nodes $\Delta = P - (2m - 1)$ is fixed. Our condition number bound may be viewed as a result of independent interest in the area of numerical methods, and requires nontrivial use of techniques from numerical approximation theory. This result is in contrast with the well known exponential growth for Vandermonde systems. We also show the significant improvement in stability via numerical experiments in Section V-C. We also provide a preview of the results here in Table I, whose results demonstrate that remarkably, our Chebyshev-Vandermonde construction with even $P = 150$ nodes has a smaller relative error than the Vandermonde-based MatDot Codes[4] with $P = 30$ nodes.

[4]We note that the numerical error depends not only on the condition number of the matrix, but also the algorithm used for solving the linear system. However, we are not aware of any approach that can accurately solve, say, a 150×150 linear system with a Vandermonde matrix (See e.g., [24], [25]).

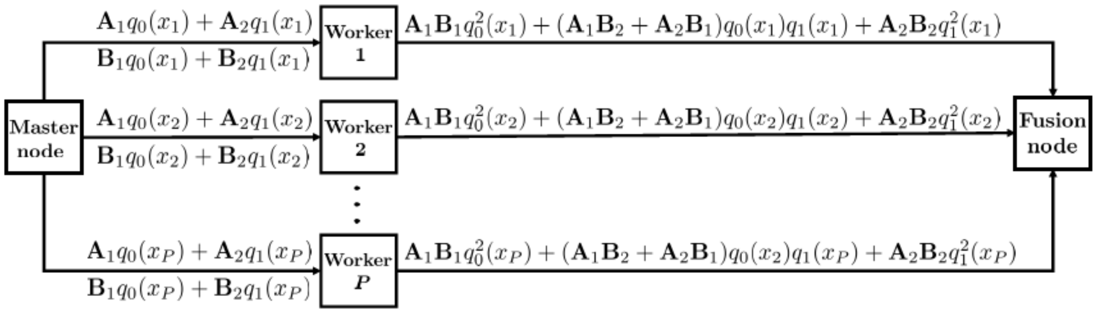

$\mathbf{A}_1q_0(x_1) + \mathbf{A}_2q_1(x_1)$
$\mathbf{B}_1q_0(x_1) + \mathbf{B}_2q_1(x_1)$
Worker 1
$\mathbf{A}_1\mathbf{B}_1q_0^2(x_1) + (\mathbf{A}_1\mathbf{B}_2 + \mathbf{A}_2\mathbf{B}_1)q_0(x_1)q_1(x_1) + \mathbf{A}_2\mathbf{B}_2q_1^2(x_1)$
Master node
$\mathbf{A}_1q_0(x_2) + \mathbf{A}_2q_1(x_2)$
$\mathbf{B}_1q_0(x_2) + \mathbf{B}_2q_1(x_2)$
Worker 2
$\mathbf{A}_1\mathbf{B}_1q_0^2(x_2) + (\mathbf{A}_1\mathbf{B}_2 + \mathbf{A}_2\mathbf{B}_1)q_0(x_2)q_1(x_2) + \mathbf{A}_2\mathbf{B}_2q_1^2(x_2)$
Fusion node
$\mathbf{A}_1q_0(x_P) + \mathbf{A}_2q_1(x_P)$
$\mathbf{B}_1q_0(x_P) + \mathbf{B}_2q_1(x_P)$
Worker P
$\mathbf{A}_1\mathbf{B}_1q_0^2(x_P) + (\mathbf{A}_1\mathbf{B}_2 + \mathbf{A}_2\mathbf{B}_1)q_0(x_2)q_1(x_P) + \mathbf{A}_2\mathbf{B}_2q_1^2(x_P)$

Fig. 5. Example of our proposed orthonormal polynomials based codes, with a recovery threshold of 3. The matrix product $\mathbf{AB}$ is $\int_{-1}^{1} p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)dx$, and can be recovered at the fusion node upon receiving the output of any 3 worker nodes, then interpolating $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$, and computing the integral $\int_{-1}^{1} p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)dx$.

TABLE I

A Table Depicting the Relative Errors of Various Schemes for $\Delta = P - (2m-1) = 3$ Redundant Nodes. The Error Is Measured via the Frobenius Norm, i.e., $\frac{||\mathbf{AB}-\hat{\mathbf{C}}||_F}{||\mathbf{AB}||_F}$. The Matrices $\mathbf{A}, \mathbf{B}$ Are Chosen With Entries $\mathcal{N}(0,1)$. The Average Relative Error Averaged Over All Possible 3 Node Failures, i.e., Over Every Set of $2m-1$ Nodes Among the $P = 2m+2$ Nodes; the Worst Case Relative Error Involves the Worst Set of $2m-1$ Nodes. See Section V-C for More Details.

Number of Workers (P)	MatDot worst case relative error	OrthoMatDot worst case relative error	MatDot average relative error	OrthoMatDot average relative error
30	1.54×10^{-6}	5.14×10^{-11}	1.36×10^{-7}	1.36×10^{-13}
50	8.6×10^{3}	1.27×10^{-9}	2.00×10^{2}	2.04×10^{-13}
80	2.45×10^{6}	1.98×10^{-8}	2.19×10^{2}	3.08×10^{-12}
150	3.87×10^{7}	7.84×10^{-7}	8.73×10^{2}	2.03×10^{-11}

While MatDot Codes [3] have an optimal recovery threshold of $2m-1$, they have relatively higher computation cost per worker ($O(N^3/m)$) and worker node to fusion node communication cost ($O(N^2)$) as compared to Polynomial Codes [6] which have a computation cost per worker of $O(N^3/m^2)$ and worker node to fusion node communication cost of $O(N^2/m^2)$. In particular, each worker in MatDot Codes performs an "outer" product of an $N \times N/m$ matrix with an $N/m \times N$ matrix, whereas each worker in Polynomial Codes performs an "inner" product of an $N/m \times N$ matrix with an $N \times N/m$ matrix. The reduced computation/communication comes at the cost of weaker fault-tolerance - Polynomial Codes have a higher recovery threshold of m^2 as compared with MatDot Codes ($2m-1$). In Section VI, we develop numerically stable codes for matrix multiplication, again via orthogonal polynomials, that achieve the same low computation/communication costs as Polynomial Codes as well as the same recovery threshold; we refer to these codes as *OrthoPoly Codes*.

The trade-off between computation/communication cost and recovery threshold imposed by MatDot Codes and Polynomial Codes has motivated general code constructions that interpolates both of them [3], [5], [26], albeit using the monomial basis. In Section VII, we extend our approach to a general matrix multiplication code construction, referred to as *Generalized OrthoMatDot*, that offers a computation/communication cost vs recovery threshold trade-off, following the research thread of [3], [5], [26], [27], however we also target numerical stability in our proposed construction. While our Generalized OrthoMatDot Codes specialize to OrthoMatDot Codes, i.e., they achieve the same optimal recovery threshold as OrthoMatDot Codes when allowing for the same computation/communication cost as OrthoMatDot Codes, they do not specialize to OrthoPoly Codes. Specifically, Generalized OrthoMatDot codes have higher recovery threshold than OrthoPoly Codes when allowing for the same computation/communication cost as OrthoPoly Codes. In Section VIII, we exploit the result obtained in Theorem 5.1 on the condition number of the square $K \times K$ sub-matrices of the $K \times P$ Chebyshev-Vandermonde matrices to propose a numerically stable algorithm for Lagrange coded computing. In Section IX, we conclude with a discussion on other related problems such as matrix-vector multiplication [13], [28], and describe some related open questions.

III. Preliminaries on Numerical Analysis and Notation

We discuss, in this section, the problem of finite precision in representing real numbers on digital machines and how it may horribly affect the output of computation problems performed on these machines. In addition, we also introduce some basic definitions and results from the area of numerical approximation theory that will be used in this paper [23], [29]. Next, we describe the computation environment adopted in running the numerical experiments of this paper. Finally, at the end of this section, we provide most of the common notation that will be used in this paper.

A. Preliminaries on Numerical Analysis

Since digital machines have finite memory, real numbers are digitally stored using a finite number of bits, i.e., finite precision. However, storing real numbers using a finite number of bits leads to inevitable errors since a finite number of bits can only represent a finite number of real numbers with no errors. On the other hand, real numbers that cannot be directly represented using the specified finite number of bits have to be either truncated or rounded-off in order to fit in the memory. Although such perturbation (e.g., truncation/round-off error) of real numbers due to the finite precision of digital machines can be negligibly small, the perturbation of the output of any computation, that uses such "small" perturbed stored real numbers as input, is not necessarily small as well. In fact, a very small perturbation to the input of some computation may lead to an output that is totally wrong and irrelevant to the correct output. The condition number of a computation problem captures/measures this observation.

Definition 3.1 (Condition Number): Let f be a function $\mathbb{R}^m \to \mathbb{R}^n, m, n \in \mathbb{N}^+$, representing a computation problem with input x, and let δx be a small perturbation of x, and define $\delta f(x) = f(x + \delta x) - f(x)$ to be the perturbation of f at x due to δx, the condition number of the problem at x with respect to some norm $||\cdot||$ is

$$\kappa(x) = \sup_{\delta x} \left(\frac{||\delta f(x)||}{||f(x)||} \Big/ \frac{||\delta x||}{||x||} \right). \tag{2}$$

Given the above definition of condition number, a problem is said to be "ill-conditioned" if small perturbations in the input lead to large perturbation in the output (i.e., the condition number is large). On the other hand, a problem is said to be "well-conditioned" if small perturbations in the input lead to small perturbations in the output (i.e., the condition number is small).

In what follows, we discuss the condition number of two computation problems: the matrix-vector multiplication and solving a system of linear equations. For both problems, consider the system of linear equations represented in the matrix form $\mathbf{A}\mathbf{x} = \mathbf{y}$, where $\mathbf{A} \in \mathbb{R}^{n,n}$ and nonsingular, and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, and let $||\cdot||$ be some matrix norm. Then, let $\mathbf{A}$ be fixed, the condition number of this matrix-vector multiplication problem with $\mathbf{y}$ as its output given small perturbations in the input $\mathbf{x}$ is $\kappa(\mathbf{x}) \leq ||\mathbf{A}||||\mathbf{A}^{-1}||$, for any $\mathbf{x} \in \mathbb{R}^n$. Also, for the problem of solving the system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{y}$, with $\mathbf{A}$ still fixed, the condition number of the problem of solving this system of linear equations, given small perturbations in the input $\mathbf{y}$, where $\mathbf{x}$ is the output, is $\kappa(\mathbf{y}) \leq ||\mathbf{A}||||\mathbf{A}^{-1}||$, for any $\mathbf{y} \in \mathbb{R}^n$.

Definition 3.2 (Matrix Condition Number): For any nonsingular matrix $\mathbf{A}$, the term $||\mathbf{A}||||\mathbf{A}^{-1}||$ is defined as the condition number of $\mathbf{A}$ with respect to the norm $||\cdot||$ and is denoted by $\kappa(\mathbf{A})$, i.e., $\kappa(\mathbf{A}) = ||\mathbf{A}||||\mathbf{A}^{-1}||$.

Definition 3.3 (Relative Error): Let f be a function $\mathbb{R}^m \to \mathbb{R}^n, m, n \in \mathbb{N}^+$ with input x, and let δx be a small perturbation of x, and define $\delta f(x) = f(x + \delta x) - f(x)$ to be the perturbation of f at x due to δx, the relative error at x with respect to some norm $||\cdot||$ is defined as

$$\frac{||\delta f(x)||}{||f(x)||} = \frac{||f(x + \delta x) - f(x)||}{||f(x)||}. \tag{3}$$

Next, we introduce some basic tools of numerical approximation theory that will be used throughout this paper. Notice that, in the following, $C[a, b]$ denotes the vector space of continuous integrable functions defined on the interval $[a, b]$.

Definition 3.4 (Inner Products on $C[a, b]$): For any $f, g \in C[a, b]$, and given a nonnegative integrable weight function w,

$$\langle f, g \rangle = \int_a^b f(x)g(x)w(x)dx$$

defines an inner product on $C[a, b]$ relative to w.

Definition 3.5 (Orthogonal Polynomials): Consider a nonnegative integrable weight function w, the polynomials $\{q_i\}_{i\geq 0}$ in $C[a, b]$ where $q_i(x)$ has degree i and

$$\langle q_i, q_j \rangle = \begin{cases} c_i & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

for some nonzero values c_i, where the inner product is relative to w, are called orthogonal polynomials relative to w.

Definition 3.6 (Orthonormal Polynomials): Consider a nonnegative integrable weight function w, the polynomials $\{q_i\}_{i\geq 0}$, where $q_i(x)$ has degree i, in $C[a, b]$ such that

$$\langle q_i, q_j \rangle = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where the inner product is relative to w, are called orthonormal polynomials relative to w.

Note that based on the above definitions, if the polynomials $\{q_i\}_{i\geq 0}$ are orthogonal (or orthonormal), then $q_n(x)$ is orthogonal to all polynomials of degree $\leq n-1$, i.e., $\langle p_{n-1}(x), q_n(x) \rangle = 0$, for any polynomial $p_{n-1} \in C[a, b]$ with degree strictly less than n. It's also worth noting that for $w(x) = 1, a = -1, b = 1$, the orthogonal polynomials are Legendre polynomials, which are derived via Gram-Schmidt procedure applied to $\{1, x, x^2, \ldots, \}$ sequentially. In addition, the following is an important class of orthogonal polynomials in our paper.

Example 3.1 (Chebyshev Polynomials of the First Kind): The following recurrence relation defines the Chebyshev polynomials of the first kind:

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x),$$

where, $T_0(x) = 1, T_1(x) = x$. These Chebyshev polynomials are the cornerstone of modern numerical approximation theory and practice with applications to numerical integration, and least-square approximations of continuous functions [23], [29]. $\frac{1}{\sqrt{2}}T_0, T_1, T_2, \cdots$ are orthonormal relative to the weight function $\frac{2}{\pi\sqrt{1-x^2}}$. In general, Chebyshev polynomials are defined over $x \in \mathbb{R}$. However, for $x \in [-1, 1]$, $T_n(x) = \cos(n \arccos(x))$, for any $n \in \mathbb{N}$. For the rest of this paper, unless otherwise is stated, whenever Chebyshev polynomials are used, they are restricted only to the range $[-1, 1]$.

We state, next, two results from [29] in Theorems 3.1 and 3.2.

Theorem 3.1: Let w be a weight function on the range $[a, b]$, i.e., w is a nonnegative integrable function on $[a, b]$, and let $x_1, \cdots, x_n$ be distinct real numbers such that $a < x_1 < \cdots < x_n < b$, there exist unique weights $a_1, \cdots, a_n$ such that

$$\int_a^b f(x)w(x)dx = \sum_{i=1}^{n} a_i f(x_i),$$

for all polynomials f with degree less than n.

Theorem 3.1 is not surprising - the left hand side of the equation stated in the theorem is a linear operator on the vector space of $n-1$-degree polynomials. Because of Lagrange-interpolation, the space of $n-1$-degree polynomials is itself a linear transformation on its evaluation at n points. Therefore, the left hand side can be expressed as an inner product of the functions evaluations at n points. We next state a remarkable result by Gauss which states conditions under which the expression of Theorem 3.1 is exact for polynomials of degree up to $2n-1$, even though the number of evaluation points is just n.

Theorem 3.2 (Gauss Quadrature): Fix a weight function w, and let $\{q_i\}_{i \geq 0}$ be a set of orthonormal polynomials in $C[a, b]$ relative to w. Given n, let $\eta_1, \cdots, \eta_n$ be the roots of q_n such that $a \leq \eta_1 < \eta_2 < \cdots < \eta_n \leq b$, and choose real values $a_1, \cdots, a_n$ such that $\sum_{i=1}^{n} a_i f(\eta_i) = \int_a^b f(x)w(x)dx$, for any $f \in C[a, b]$ with degree less than n. Then, $\sum_{i=1}^{n} a_i f(\eta_i) = \int_a^b f(x)w(x)dx$, for any polynomial f with degree less than $2n$.

Remark 3.1:

1) Consider any orthonormal polynomials $\{q_i\}_{i \geq 0}$. For any $n \in \mathbb{N}$, the set $\{q_0, q_1, \cdots, q_{n-1}\}$ forms a basis for the vector space of polynomials with degree less than n.
2) In Theorem 3.2, $a_1, \cdots, a_n$ can be chosen as
$$a_i = \int_a^b \left(\prod_{j \in [n]-i} \frac{x - \eta_j}{\eta_i - \eta_j} \right) w(x)dx, \ i \in [n]. \quad (6)$$
Notice that although the computation of the parameters $a_1, \cdots, a_n$ can be nontrivial in general, as we see next, their computation is simple when the set of orthonormal polynomials is restricted to Chebyshev polynomials.
3) In Theorem 3.2, the roots of q_n, i.e., $\eta_1, \cdots, \eta_n$ are, in fact, real and distinct. Moreover, the Chebyshev polynomial of the first kind T_n has the following roots
$$\rho_i^{(n)} = \cos\left(\frac{2i-1}{2n}\pi\right), \ i \in [n]. \quad (7)$$
The set $\{\rho_1^{(n)}, \cdots, \rho_n^{(n)}\}$ is often called the n-point Chebyshev grid, and its elements $\rho_1^{(n)}, \cdots, \rho_n^{(n)}$ are called "Chebyshev nodes" of degree n. We here discard the term "node" and use the term "Chebyshev points" to avoid confusion with computation nodes. We also denote by $\boldsymbol{\rho}^{(n)}$ the vector $(\rho_1^{(n)}, \cdots, \rho_n^{(n)})$. It is useful to note that $T_n(x)$ can be written as
$$T_n(x) = 2^{n-1} \prod_{i=1}^{n} (x - \rho_i^{(n)}), \quad (8)$$
and for $T_n(x)$, the parameters a_i in (6) are all equal to $2/n$ when $w(x) = \frac{2}{\pi\sqrt{1-x^2}}$ [29, Chapter 9].

Remark 3.2: Note that an ill-conditioned problem does not always mean that the numerical errors are large. In particular, for the codes studied in previous works that use the monomial basis, it is possible to improve numerical accuracy of decoding over naïve interpolation, especially if inputs and outputs have a limited range. For instance, reference [30] quantizes real values to finite fields where exact calculations can be performed; this can sometimes reduce numerical errors as compared with naïve methods (see also [6]). Since performing arithmetic operations over a finite field $\mathbb{F}_{2^n}$ (or even $\mathbb{F}_p$ where p is the largest prime number that is less than 2^n) requires representing each element as an n bit vector, this solution applies when the input ranges can be used in *fixed point* representation of n bits without losing too much accuracy. Recall that *floating* point represents a much greater range as compared with fixed point representation for the same memory (e.g., n bits). It must be noted that much care is required to make the approach of [30] work correctly. Not only must the quantization error be small enough in representing the input, but also the value of n must be large enough so that no wrapping-around happens when computations are performed in the finite field. For instance, when 64-bit computing is applied, for reasons described in [30], n can be chosen to be 24. As a simple example, assume that matrix multiplication is performed between two matrices of the same range, to prevent wrapping around, at most 12 bits can be used for each multiplicand, which lower bounds the overall error to a fraction of $1/2^{12} \approx 0.25 \times 10^{-3}$ the range of the input. The error can be even larger depending on the dimension of the matrix and the number of nodes. Such an approach would be particularly handicapped in exploiting the power of floating point computations, where, for example in the 64-bit IEEE Standard for Floating Point Arithmetic 754, inputs ranging from 10^{-318} to 10^{318} (in absolute value) are represented. Indeed, for this reason, it is standard practice in numerical analysis [20], as well as our paper, to study the condition number as a mathematical formulation that is algorithm/implementation-independent. A second drawback of the approach of [30] is that it requires control of the matrix multiplication algorithm used in the worker nodes, our approach is universal; we only require control of the encoding and decoding, and the worker nodes can use any black-box matrix multiplication algorithm.

We show novel code constructions and show that they are numerically stable (i.e., lead to small numerical errors) in an algorithm independent manner. Specifically, we prove formal bounds for condition numbers. Building on the condition number bounds, we conduct a full, formal, forward error analysis of our entire system (including encoding and decoding) in Appendix D, showing that applying the proposed code construction with the well-conditioned decoding matrix (specifically when the condition number of the decoding matrix is much smaller than the inverse of the machine precision) yields an output with relatively small numerical errors.

B. Computation Environment

All the numerical experiments in this paper were conducted using MATLAB [31] with double precision floating point numbers system with machine epsilon $\approx 10^{-16}$ according to the IEEE Standard for Floating Point Arithmetic 754. The programs used for testing our code constructions are accessible through the repository in [32].

C. Notation

Throughout this paper, we use lowercase bold letters to denote vectors and uppercase bold letters to denote matrices. In addition, for any positive integers k, n, and given a set of orthogonal polynomials $q_0, q_1, \cdots, q_{k-1}$ on the interval $[a, b]$, let $\mathbf{x} = (x_1, \cdots, x_n)$ be a vector with entries in $[a, b]$, we define the $k \times n$ matrix $\mathbf{Q}^{(k,n)}(\mathbf{x})$ as:

$$\mathbf{Q}^{(k,n)}(\mathbf{x}) = \begin{pmatrix} q_0(x_1) & \cdots & q_0(x_n) \\ \vdots & \ddots & \vdots \\ q_{k-1}(x_1) & \cdots & q_{k-1}(x_n) \end{pmatrix}. \tag{9}$$

For any subset $\mathcal{S} = \{s_1, \cdots, s_r\} \subset [n]$, we denote by $\mathbf{Q}_{\mathcal{S}}^{(k,n)}(\mathbf{x})$ the sub-matrix of $\mathbf{Q}^{(k,n)}(\mathbf{x})$ formed by concatenating columns with indices in $\mathcal{S}$, i.e.,

$$\mathbf{Q}_{\mathcal{S}}^{(k,n)}(\mathbf{x}) = \begin{pmatrix} q_0(x_{s_1}) & \cdots & q_0(x_{s_r}) \\ \vdots & \ddots & \vdots \\ q_{k-1}(x_{s_1}) & \cdots & q_{k-1}(x_{s_r}) \end{pmatrix}. \tag{10}$$

For the special case where the orthogonal polynomials are the Chebyshev polynomials of the first kind $T_0, T_1, \cdots, T_{k-1}$, we define the $k \times n$ matrix $\mathbf{G}^{(k,n)}(\mathbf{x})$ as:

$$\mathbf{G}^{(k,n)}(\mathbf{x}) = \begin{pmatrix} T_0(x_1) & \cdots & T_0(x_n) \\ \vdots & \ddots & \vdots \\ T_{k-1}(x_1) & \cdots & T_{k-1}(x_n) \end{pmatrix}, \tag{11}$$

we denote by $\mathbf{G}_{\mathcal{S}}^{(k,n)}(\mathbf{x})$ the sub-matrix of $\mathbf{G}^{(k,n)}(\mathbf{x})$ formed by concatenating columns with indices in $\mathcal{S}$, i.e.,

$$\mathbf{G}_{\mathcal{S}}^{(k,n)}(\mathbf{x}) = \begin{pmatrix} T_0(x_{s_1}) & \cdots & T_0(x_{s_r}) \\ \vdots & \ddots & \vdots \\ T_{k-1}(x_{s_1}) & \cdots & T_{k-1}(x_{s_r}) \end{pmatrix}. \tag{12}$$

Also, for the case where the orthogonal polynomials are the "orthonormal" Chebyshev polynomials $\frac{1}{\sqrt{2}}T_0, T_1, \cdots, T_{k-1}$, we define the $k \times n$ matrix $\tilde{\mathbf{G}}^{(k,n)}(\mathbf{x})$ as:

$$\tilde{\mathbf{G}}^{(k,n)}(\mathbf{x}) = \begin{pmatrix} T_0(x_1)/\sqrt{2} & \cdots & T_0(x_n)/\sqrt{2} \\ T_1(x_1) & \cdots & T_1(x_n) \\ \vdots & \ddots & \vdots \\ T_{k-1}(x_1) & \cdots & T_{k-1}(x_n) \end{pmatrix}, \tag{13}$$

and we denote by $\tilde{\mathbf{G}}_{\mathcal{S}}^{(k,n)}(\mathbf{x})$ the sub-matrix of $\tilde{\mathbf{G}}^{(k,n)}(\mathbf{x})$ formed by concatenating columns with indices in $\mathcal{S}$, i.e.,

$$\tilde{\mathbf{G}}_{\mathcal{S}}^{(k,n)}(\mathbf{x}) = \begin{pmatrix} T_0(x_{s_1})/\sqrt{2} & \cdots & T_0(x_{s_r})/\sqrt{2} \\ T_1(x_{s_1}) & \cdots & T_1(x_{s_r}) \\ \vdots & \ddots & \vdots \\ T_{k-1}(x_{s_1}) & \cdots & T_{k-1}(x_{s_r}) \end{pmatrix}. \tag{14}$$

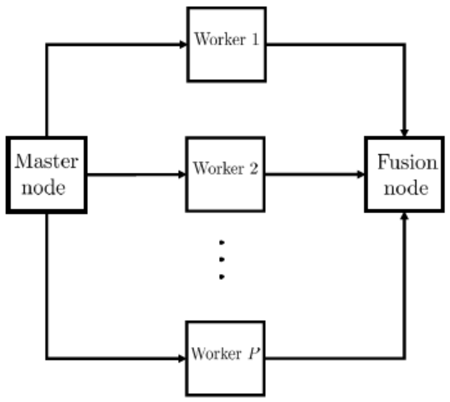


Fig. 6. The distributed system framework.

Wherever there is no ambiguity on $\mathbf{x}$, it may be dropped from the notation.

In the next section, we show that orthonormal polynomials can be used for designing codes for the distributed large scale matrix multiplication problem.

IV. OrthoMatDot: Orthonormal Polynomials Based Codes for Distributed Matrix Multiplication

In this section, we present a new orthonormal-polynomials-based class of codes for matrix multiplication called OrthoMatDot. These codes achieve the same recovery threshold as MatDot Codes, and have similar computational complexity as MatDot. The main advantage of the proposed codes is that they avoid dealing with the ill-conditioned monomial basis used in previous work (e.g., in [3], [5], [6], [26]). In Section V, OrthoMatDot Codes will be specialized and demonstrated to have higher numerical stability as compared with the state of the art. We begin with a formal problem formulation in Section IV-A, and describe our codes in Section IV-B.

A. System Model and Problem Formulation

1) System Model: We consider the distributed framework depicted in Fig. 6 that consists of a master node, P worker nodes, and a fusion node where the only communication allowed is from the master node to the different worker nodes and from the worker nodes to the fusion node. It can happen that the fusion node and the master node be represented by the same node. In this case, the only communication allowed is the communication between the master node and every worker node.

2) Problem Formulation: The master node possesses two real-valued input matrices $\mathbf{A}$, $\mathbf{B}$ with dimensions $N_1 \times N_2$, $N_2 \times N_3$, respectively. Every worker node receives from the master node an encoded matrix of $\mathbf{A}$ of dimension $N_1 \times N_2/m$ and an encoded matrix of $\mathbf{B}$ of dimension $N_2/m \times N_3$, and performs matrix multiplication of these two received inputs. Upon performing the matrix multiplication, each worker node sends the result to the fusion node. The fusion node needs to recover the matrix multiplication $\mathbf{AB}$ once it receives the results of any K worker nodes, where $K \leq P$. In this case, K is denoted by *the recovery threshold* of the distributed computing scheme.

B. OrthoMatDot Code Construction

Our result regarding the existence of achievable codes solving the distributed matrix multiplication problem using orthonormal polynomials is stated in the following theorem.

Theorem 4.1: For the matrix multiplication problem described in Section IV-A2 computed on the system defined in Section IV-A1, a recovery threshold of $2m-1$ is achievable using any set of orthonormal polynomials $\{q_i\}_{i\geq 0}$ relative to some weight polynomial w and defined on a range $[a,b]$.

Before proving this theorem, we first present OrthoMatDot, a code construction that achieves the recovery threshold of $2m-1$ given any set $\{q_i\}_{i\geq 0}$ of orthonormal polynomials relative to a weight polynomial $w(x)$ and defined on a range $[a,b]$. In our code construction, we assume that matrix $\mathbf{A}$ is split vertically into m equal sub-matrices, of dimension $N_1 \times N_2/m$ each, and matrix $\mathbf{B}$ is split horizontally into m equal sub-matrices, of dimension $N_2/m \times N_3$ each, as follows:

$$\mathbf{A} = (\mathbf{A}_0 \; \mathbf{A}_1 \; \ldots \; \mathbf{A}_{m-1}), \quad \mathbf{B} = \begin{pmatrix} \mathbf{B}_0 \\ \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_{m-1} \end{pmatrix}, \tag{15}$$

we also define a set of P distinct real numbers $x_1, \cdots, x_P$ in the range $[a,b]$, and define two encoding polynomials $p_{\mathbf{A}}(x) = \sum_{i=0}^{m-1} \mathbf{A}_i q_i(x)$ and $p_{\mathbf{B}}(x) = \sum_{i=0}^{m-1} \mathbf{B}_i q_i(x)$, and let $p_{\mathbf{C}}(x) = p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$.

In the following, we briefly describe the OrthoMatDot construction. First, for every $r \in [P]$, the master node sends to the r-th worker node evaluations of $p_{\mathbf{A}}(x), p_{\mathbf{B}}(x)$ at $x = x_r$, that is, it sends $p_{\mathbf{A}}(x_r)$ and $p_{\mathbf{B}}(x_r)$ to the r-th worker node. Next, for every $r \in [P]$, the r-th worker node computes the matrix product $p_{\mathbf{C}}(x_r) = p_{\mathbf{A}}(x_r)p_{\mathbf{B}}(x_r)$ and sends the result to the fusion node. Once the fusion node receives the output of any $2m-1$ worker nodes, it interpolates the polynomial $p_{\mathbf{C}}(x) = p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ with respect to the orthonormal basis $\{q_i\}_{i\geq 0}$ (recall the first point in Remark 3.1) and evaluates $p_{\mathbf{C}}(x)$ at $\eta_1, \cdots, \eta_m$, where $\eta_1, \cdots, \eta_m$ are the roots of q_m. Then, it performs the summation $\sum_{r=1}^{m} a_r \, p_{\mathbf{C}}(\eta_r)$, where $a_1, \cdots, a_m$ are as in (6).

We formally present OrthoMatDot code in Construction 1. Construction 1 uses the following notation: The output of the computation system is the $N_1 \times N_3$ matrix $\hat{\mathbf{C}}$. The (i,j)-th entries of the matrix polynomial $p_{\mathbf{C}}(x)$ and the matrix $\hat{\mathbf{C}}$ are respectively denoted as $p_{\mathbf{C}}^{(i,j)}(x)$ and $\hat{C}(i,j)$. The reader may also recall the definition of matrices $\mathbf{Q}^{(2m-1,P)}(\mathbf{x})$ and $\mathbf{Q}_{\mathcal{R}}^{(2m-1,P)}(\mathbf{x})$, for any subset $\mathcal{R} = \{r_1, \cdots, r_{2m-1}\} \subset [P]$. $\boldsymbol{\eta} = (\eta_1, \cdots, \eta_m)$ is the vector of the roots of q_m. Based on Construction 1, we state the following claim.

Claim 4.2: $\mathbf{AB} = \sum_{r=1}^{m} a_r \, p_{\mathbf{C}}(\eta_r)$.

The proof of Claim 4.2 is provided in Appendix A.

Now, we can prove Theorem 4.1.

Proof of Theorem 4.1: In order to prove the theorem, it suffices to show that Construction 1 is a valid construction with a recovery threshold of $2m-1$. Therefore, in the following, we prove that Construction 1 can recover $\mathbf{AB}$ after the fusion node receives the output of at most $2m-1$ worker nodes. Assume that the fusion node has already received the results of any $2m-1$ worker nodes. Now, because the polynomial $p_{\mathbf{C}}(x)$ has degree $2m-2$, the evaluations of $p_{\mathbf{C}}(x)$ at any $2m-1$ distinct points is sufficient to interpolate the polynomial, and since $x_1, \cdots, x_P$ are distinct, the fusion node can interpolate $p_{\mathbf{C}}(x)$ once it receives the output of any $2m-1$ worker nodes. Afterwards, given that $\mathbf{AB} = \sum_{r=1}^{m} a_r \, p_{\mathbf{C}}(\eta_r)$ (Claim 4.2), the fusion node can evaluate $p_{\mathbf{C}}(\eta_1), \cdots, p_{\mathbf{C}}(\eta_m)$ and perform the scaled summation $\sum_{r=1}^{m} a_r \, p_{\mathbf{C}}(\eta_r)$ to recover $\mathbf{AB}$. □

Construction 1 OrthoMatDot: Inputs: $\mathbf{A}, \mathbf{B}$, Output: $\hat{\mathbf{C}}$

```
1: procedure MasterNode($\mathbf{A}, \mathbf{B}$)    ▷ The master node's procedure
2:     $r \leftarrow 1$
3:     while $r \neq P+1$ do
4:         $p_{\mathbf{A}}(x_r) \leftarrow \sum_{i=0}^{m-1} \mathbf{A}_i q_i(x_r)$
5:         $p_{\mathbf{B}}(x_r) \leftarrow \sum_{i=0}^{m-1} \mathbf{B}_i q_i(x_r)$
6:         send $p_{\mathbf{A}}(x_r), p_{\mathbf{B}}(x_r)$ to worker node $r$
7:         $r \leftarrow r+1$
8:     end while
9: end procedure
10:
11: procedure WorkerNode($p_{\mathbf{A}}(x_r), p_{\mathbf{B}}(x_r)$)    ▷ The procedure of worker node $r$
12:     $p_{\mathbf{C}}(x_r) \leftarrow p_{\mathbf{A}}(x_r)p_{\mathbf{B}}(x_r)$
13:     send $p_{\mathbf{C}}(x_r)$ to the fusion node
14: end procedure
15:
16: procedure FusionNode($\{p_{\mathbf{C}}(x_{r_1}), \cdots, p_{\mathbf{C}}(x_{r_{2m-1}})\}$) ▷ The fusion node's procedure, $r_i$'s are distinct
17:     $\mathbf{Q}_{\text{inv}} \leftarrow \left(\mathbf{Q}_{\mathcal{R}}^{(2m-1,P)}\right)^{-1}$
18:     for $i \in [N_1]$ do
19:         for $j \in [N_3]$ do
20:             $(c_0^{(i,j)}, \cdots, c_{2m-2}^{(i,j)}) \leftarrow$
                    $(p_{\mathbf{C}}^{(i,j)}(x_{r_1}), \cdots, p_{\mathbf{C}}^{(i,j)}(x_{r_{2m-1}}))\mathbf{Q}_{\text{inv}}$
21:             $(p_{\mathbf{C}}^{(i,j)}(\eta_1), \cdots, p_{\mathbf{C}}^{(i,j)}(\eta_m)) \leftarrow$
                    $(c_0^{(i,j)}, \cdots, c_{2m-2}^{(i,j)})\mathbf{Q}^{(2m-1,m)}(\boldsymbol{\eta})$
22:             $\hat{C}(i,j) \leftarrow$
                    $(p_{\mathbf{C}}^{(i,j)}(\eta_1), \cdots, p_{\mathbf{C}}^{(i,j)}(\eta_m))(a_1, \cdots, a_m)^T$    ▷ $a_i$'s are as defined in (6)
23:         end for
24:     end for
25:     return $\hat{\mathbf{C}}$
26: end procedure
```

Remark 4.1: In Construction 1, setting $x_1, \cdots, x_m$ to be the roots of q_m leads to a faster decoding for the scenarios in which the first m worker nodes send their results but only less than $2m-1$ workers succeed to send their outputs. For such scenarios, we have $\sum_{r=1}^{m} a_r \, p_{\mathbf{C}}(x_r) = \sum_{r=1}^{m} a_r \, p_{\mathbf{C}}(\eta_r) = \mathbf{AB}$, where the last equality follows from Claim 4.2.

Next, we study the computational and communication costs of OrthoMatDot.

1) Complexity Analyses of OrthoMatDot:

Encoding Complexity: Encoding for each worker requires performing two additions, each adding m scaled matrices of size $N_1\ N_2/m$ and $N_2\ N_3/m$, for an overall encoding complexity for each worker of $O(N_1\ N_2 + N_2N_3)$. Therefore, the overall computational complexity of encoding for P workers is $O(N_1N_2P + N_2N_3P)$.

Computational Cost per Worker: Each worker multiplies two matrices of dimensions $N_1 \times N_2/m$ and $N_2/m \times N_3$, requiring $O(N_1N_2N_3/m)$ operations.

Decoding Complexity: Since $p_{\mathbf{C}}(x)$ has degree $2m-2$, the interpolation of $p_{\mathbf{C}}(x)$ requires the inversion of a $(2m-1)\times(2m-1)$ matrix, with complexity $O(m^3)$, and performing N_1N_3 matrix-vector multiplications, each of them is between the inverted matrix and a column vector of length $2m-1$ of the received evaluations of the matrix polynomial $p_{\mathbf{C}}(x)$ at some position $(i,j) \in [N_1]\times[N_3]$, with complexity $O(N_1N_3m^2)$. Next, the evaluation of the polynomial $p_{\mathbf{C}}(x)$ at $\eta_1,\cdots,\eta_m$ requires a complexity of $O(N_1N_3m^2)$. Finally, performing the summation $\sum_{r=1}^{m} a_r p_{\mathbf{C}}(\eta_r)$ requires a complexity of $O(N_1N_3m)$. Thus, assuming that $m \ll N_1, N_3$, the overall decoding complexity is $O(m^3 + 2N_1N_3m^2 + N_1N_3m) = O(N_1N_3m^2)$.

Communication Cost: The master node sends $O(N_1N_2P/m + N_2N_3P/m)$ symbols, and the fusion node receives $O(N_1\ N_3\ m)$ symbols from the successful worker nodes.

Remark 4.2: With the reasonable assumption that the dimensions of the input matrices $\mathbf{A},\mathbf{B}$ are large enough such that $N_1,N_2,N_3 \gg m,P$, we can conclude that the encoding and decoding costs at the master and fusion nodes, respectively, are negligible compared to the computation cost at each worker node.

V. Numerically Stable Codes for Matrix Multiplication via OrthoMatDot Codes With Chebyshev Polynomials

In this section, we specialize OrthoMatDot Codes by restricting the orthonormal polynomials to be Chebyshev polynomials of the first kind $\{T_i\}_{i\geq 0}$ with the evaluation points chosen to be the P-dimensional Chebyshev grid, i.e., $x_i = \rho_i^{(P)}, i \in [P]$. Our specialized OrthoMatDot, described in Construction 2 in Section V-A, develops a decoding that involves inversion of a $(2m-1)\times(2m-1)$ sub-matrix of a $(2m-1)\times P$ Chebyshev-Vandermonde matrix. One of the main technical results of this section (and paper), presented in Theorem 5.1 in Section V-B, is an upper bound to the worst case condition number over all possible $(2m-1)\times(2m-1)$ sub-matrices of the $(2m-1)\times P$ Chebeshev-Vandermonde matrix for the case where the distinct evaluation points $x_1,\cdots,x_P$ are chosen as the Chebyshev points of degree P, i.e., $x_i = \rho_i^{(P)}, i \in [P]$. In fact, the derived bound shows that the worst case condition number grows at most polynomially in P at a fixed number of straggler/parity worker nodes. This is in contrast with the monomial basis codes where the condition number grows exponentially in P, even when there is no redundancy [16]–[19]. For a fair comparison, we compare our OrthoMatDot Codes with the current codes with the same input partitioning, i.e., MatDot Codes [3], through numerical experiments in Section V-C. We show through the numerical experiments that our proposed codes provide significantly lower numerical errors as compared to MatDot Codes.

A. Chebyshev Polynomials Based OrthoMatDot Code Construction

Recalling from Example 3.1 that $\frac{1}{\sqrt{2}}T_0, T_1, T_2, \cdots$ form an orthonormal polynomial set relative to the weight function $w(x) = \frac{2}{\pi\sqrt{1-x^2}}$, in Construction 2, we explain the application of Chebyshev polynomials of the first kind to Construction 1. Note that, in Construction 2, we assume that the input matrices $\mathbf{A}$ and $\mathbf{B}$ are also split as in (15), and let $x_1, x_2, \ldots, x_P$ be distinct real numbers in the range $[-1,1]$, and define the encoding functions $p_{\mathbf{A}}(x), p_{\mathbf{B}}(x)$ as $p_{\mathbf{A}}(x) = \frac{1}{\sqrt{2}}\mathbf{A}_0\ T_0(x) + \sum_{i=1}^{m-1}\mathbf{A}_iT_i(x)$ and $p_{\mathbf{B}}(x) = \frac{1}{\sqrt{2}}\mathbf{B}_0\ T_0(x) + \sum_{i=1}^{m-1}\mathbf{B}_iT_i(x)$, and let $p_{\mathbf{C}}(x) = p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$. The idea of our Chebyshev polynomials based OrthoMatDot code is as follows: First, for every $r \in [P]$, the master node sends to the r-th worker node $p_{\mathbf{A}}(\rho_r^{(P)})$ and $p_{\mathbf{B}}(\rho_r^{(P)})$. Next, for every $r \in [P]$, the r-th worker node computes the matrix product $p_{\mathbf{C}}(\rho_r^{(P)}) = p_{\mathbf{A}}(\rho_r^{(P)})p_{\mathbf{B}}(\rho_r^{(P)})$ and sends the result to the fusion node. Once the fusion node receives the output of any $2m-1$ worker nodes, it interpolates $p_{\mathbf{C}}(x)$. Note that this interpolation is performed with respect to the Chebyshev basis $1/\sqrt{2}\ T_0, T_1, T_2, \cdots$, that is, coefficients $c_0^{(i,j)}, c_1^{(i,j)}, \ldots, c_{2m-2}^{(i,j)}$ (in Construction 2) are found such that $p_{\mathbf{C}}^{(i,j)}(x) = \frac{1}{\sqrt{2}}c_0^{(i,j)}T_0(x) + \sum_{l=1}^{2m-2} c_l^{(i,j)}T_l(x)$ for $(i,j) \in [N_1]\times[N_3]$. Then, it evaluates $p_{\mathbf{C}}(x)$ at $\rho_1^{(m)},\cdots,\rho_m^{(m)}$, where $\rho_i^{(m)}$'s are as defined in (7), and computes $\sum_{i=1}^{m} a_i\, p_{\mathbf{C}}(\rho_i^{(m)})$, where $a_i = 2/m, i \in [m]$ based on 3) in Remark 3.1.

A formal description of our Chebyshev polynomials based OrthoMatDot code is provided in Construction 2. Construction 2 uses the following notation. We let the (i,j)-th entry of the matrix polynomial $p_{\mathbf{C}}(x)$ be denoted $p_{\mathbf{C}}^{(i,j)}(x)$ and written as $p_{\mathbf{C}}^{(i,j)}(x) = \frac{1}{\sqrt{2}}c_0^{(i,j)}T_0(x) + \sum_{l=1}^{2m-2} c_l^{(i,j)}T_l(x)$. Also, following the notation in Section III-C, we define the Chebyshev-Vandermonde matrices $\tilde{\mathbf{G}}^{(2m-1,P)}(\rho^{(P)})$, and $\tilde{\mathbf{G}}_{\mathcal{R}}^{(2m-1,P)}(\rho^{(P)})$, for any subset $\mathcal{R} = \{r_1,\cdots,r_{2m-1}\} \subset [P]$, we also define the matrix $\tilde{\mathbf{G}}^{(2m-1,m)}(\rho^{(m)})$. Finally, we assume that our construction returns an $N_1 \times N_3$ matrix $\hat{\mathbf{C}}$ representing the result of the product $\mathbf{AB}$, where the (i,j)-th entry of $\hat{\mathbf{C}}$ is $\hat{C}(i,j)$.

1) Complexity Analyses: The different encoding complexity, computational complexity per worker, decoding complexity and communication cost for Chebyshev polynomials based OrthoMatDot are the same as their counterparts of OrthoMatDot stated in Section IV-B1.

B. Evaluation Points and Condition Number Bound

When there is no redundancy, i.e., $n = 2m-1$, it is well known that the $n \times n$ decoding matrix $\mathbf{G}^{(n,n)}$ has condition

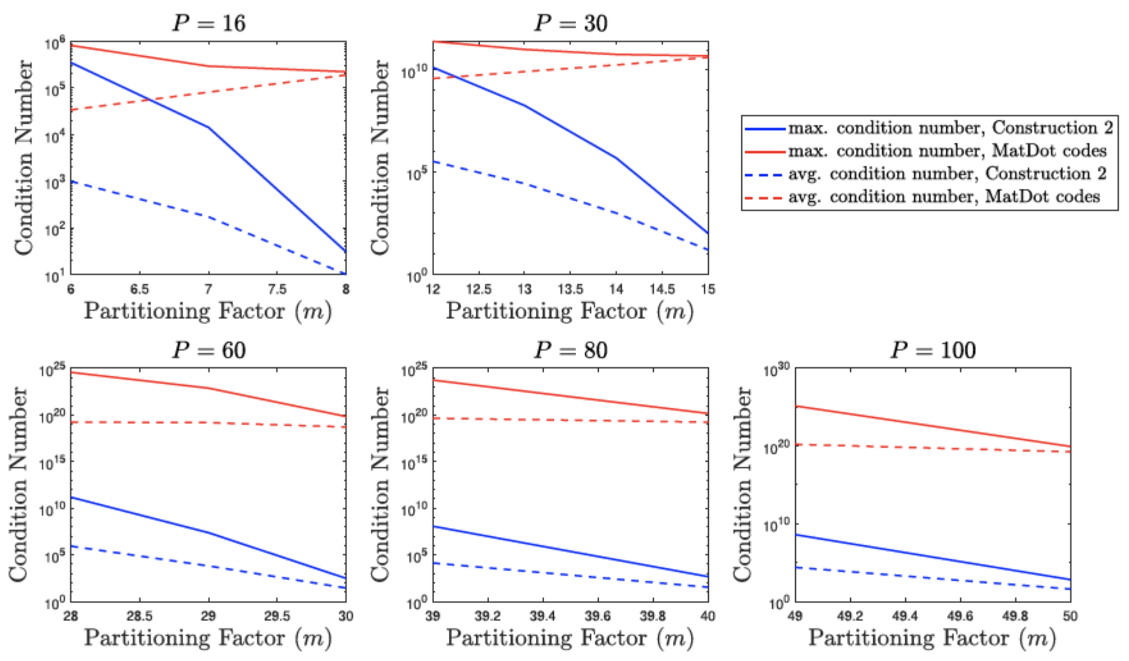


Fig. 7. Comparison between the condition number of the interpolating matrix of the Chebyshev polynomials based OrthoMatDot Codes and MatDot Codes, both with Chebyshev points as evaluation points, in five different distributed systems with $16, 30, 60, 80$, and 100 worker nodes, respectively.

number n with the ℓ_2 as well as the Frobenius norms [17]. Note the remarkable contrast with the Vandermonde matrix, whose condition number for real-valued evaluation points grows exponentially in n, no matter how the nodes are chosen [16], [17]. Our problem differs from the standard problem in numerical methods, since we have to choose a rectangular "generator" matrix where every square sub-matrix is well-conditioned. In particular, even for Chebyshev-Vandermonde matrix, if the evaluation points are not chosen carefully, they are poorly conditioned [19] (also see Fig. 8). Here, we show that choosing $x_i = \rho_i^{(n)}$ leads to a well-conditioned system with s redundant nodes. Our goal is to choose vector $\mathbf{x}$ such that $\kappa^{max}(\mathbf{G}^{(n-s,n)}(\mathbf{x}))$ is sufficiently small, where $\kappa^{max}(\mathbf{G}^{(n-s,n)}(\mathbf{x}))$ denotes the worst case condition number over all possible $(n-s)\times(n-s)$ sub-matrices of $\mathbf{G}^{(n-s,n)}(\mathbf{x})$.

Theorem 5.1: For any $s \in [n-1]$,

$$\kappa_F^{max}(\mathbf{G}^{(n-s,n)}(\rho^{(n)})) = O\left((n-s)\sqrt{ns(n-s)}\left(2n^2\right)^{s-1}\right),$$

where κ_F^{max} denotes the worst case condition number over all possible $(n-s)\times(n-s)$ sub-matrices of $\mathbf{G}^{(n-s,n)}(\mathbf{x})$ with respect to the Frobenius norm, $\rho^{(n)} = (\rho_1^{(n)}, \rho_2^{(n)}, \dots, \rho_n^{(n)})$ are the roots of the Chebyshev polynomial T_n, i.e., $\rho_i^{(n)} = \cos\left(\frac{2i-1}{2n}\pi\right), i \in [n]$.

Since $||.||_2 \leq ||.||_F$, the above bound applies to the standard ℓ_2 matrix norm as well. The proof uses techniques from numerical methods, and is provided in Appendix B.

Remark 5.1: Although the bound in Theorem 5.1 is derived for $\mathbf{G}^{(n-s,n)}(\rho^{(n)})$, the theorem also applies for $\tilde{\mathbf{G}}^{(n-s,n)}(\rho^{(n)})$. This is because it can be shown using simple matrix operations that for any $\tilde{\mathbf{G}}_{\mathcal{R}}^{(n-s,n)}$, for a subset $\mathcal{R} \subset [n]$ such that $|\mathcal{R}| = n-s$, $\kappa_F(\tilde{\mathbf{G}}_{\mathcal{R}}^{(n-s,n)}) < \sqrt{2}\,\kappa_F(\mathbf{G}_{\mathcal{R}}^{(n-s,n)})$.

C. Numerical Results

Our experiments were performed on the computation environment described in Section III-B, and the codes used in our experiments can be obtained from [32]. The numerical stability of our codes is determined by the condition number of $(2m-1)\times(2m-1)$ sub-matrices of $\mathbf{G}^{(2m-1,P)}$. The natural comparison is with MatDot Codes where the decoding depends on effectively inverting $(2m-1)\times(2m-1)$ square sub-matrices of

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_P \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{2m-2} & x_2^{2m-2} & \cdots & x_P^{2m-2} \end{pmatrix}. \quad (16)$$

Based on the result of Theorem 5.1, we choose $x_i = \rho_i^{(P)}$. In our experiments, we consider systems with various number of worker nodes, namely, $P = 16, 30, 60, 80, 100$. We compare $\kappa_2^{max}(\mathbf{G}^{(2m-1,P)})$ with $\kappa_2^{max}(\mathbf{M})$ with the Chebyshev points $\rho^{(P)}$ as evaluation points on each. We also compare the average ℓ_2 condition number of all $(2m-1)\times(2m-1)$ sub-matrices of $\mathbf{G}^{(2m-1,P)}$ and all $(2m-1)\times(2m-1)$ sub-matrices of $\mathbf{M}$. The results, in Fig. 7, show that, for every examined system, the maximum and average condition numbers of the $(2m-1)\times(2m-1)$ sub-matrices of $\mathbf{G}^{(2m-1,P)}$ are less than its MatDot Codes counterparts, especially for larger systems with $60, 80$, and 100 worker nodes. In fact, for these specific systems, the improvement in the condition number is around a scaling of 10^{15}. Due to the direct relation between the partitioning factor m and the maximum number of stragglers s to be tolerated at any fixed total number of workers P, i.e., $s = P-2m+1$, Fig. 7 also shows the relation between the condition number and the maximum number of stragglers to be tolerated at each system, e.g., for the system with 60

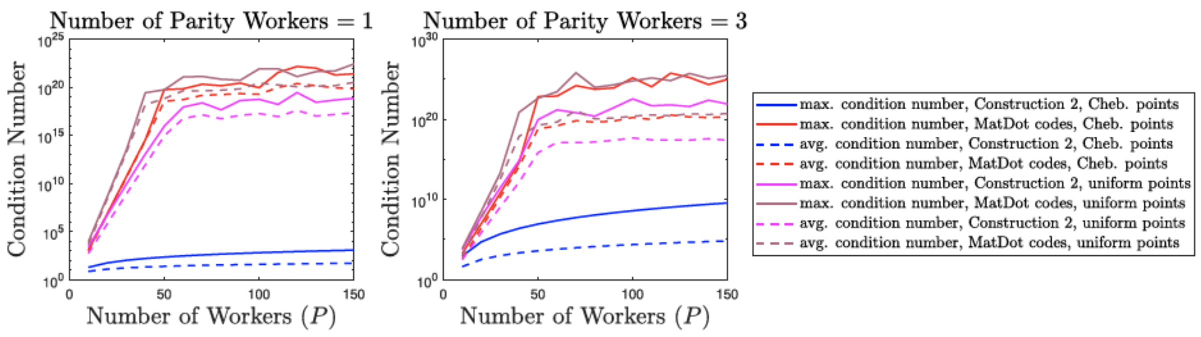


Fig. 8. The growth of the condition number, for both Chebyshev polynomials based OrthoMatDot and MatDot Codes, with the system size given a fixed number of redundant worker nodes.

Construction 2 Chebyshev Polynomials Based OrthoMatDot: **Inputs**: $\mathbf{A}, \mathbf{B}$, **Output**: $\hat{\mathbf{C}}$

```
1: procedure MASTERNODE($\mathbf{A}, \mathbf{B}$)    ▷ The master node's procedure
2:     $r \leftarrow 1$
3:     while $r \neq P+1$ do
4:         $p_{\mathbf{A}}(\rho_r^{(P)}) \leftarrow \frac{1}{\sqrt{2}}\mathbf{A}_0 + \sum_{i=1}^{m-1} \mathbf{A}_i T_i(\rho_r^{(P)})$
5:         $p_{\mathbf{B}}(\rho_r^{(P)}) \leftarrow \frac{1}{\sqrt{2}}\mathbf{B}_0 + \sum_{i=1}^{m-1} \mathbf{B}_i T_i(\rho_r^{(P)})$
6:         send $p_{\mathbf{A}}(\rho_r^{(P)}), p_{\mathbf{B}}(\rho_r^{(P)})$ to worker node $r$
7:         $r \leftarrow r+1$
8:     end while
9: end procedure
10:
11: procedure WORKERNODE($p_{\mathbf{A}}(\rho_r^{(P)}), p_{\mathbf{B}}(\rho_r^{(P)})$)    ▷ The procedure of worker node $r$
12:     $p_{\mathbf{C}}(\rho_r^{(P)}) \leftarrow p_{\mathbf{A}}(\rho_r^{(P)}) p_{\mathbf{B}}(\rho_r^{(P)})$
13:     send $p_{\mathbf{C}}(\rho_r^{(P)})$ to the fusion node
14: end procedure
15:
16: procedure FUSIONNODE($\{p_{\mathbf{C}}(\rho_{r_1}^{(P)}), \cdots, p_{\mathbf{C}}(\rho_{r_{2m-1}}^{(P)})\}$)▷ The fusion node's procedure, $r_i$'s are distinct
17:     $\mathbf{G}_{\text{inv}} \leftarrow \left(\tilde{\mathbf{G}}_{\mathcal{R}}^{(2m-1,P)}\right)^{-1}$
18:     for $i \in [N_1]$ do
19:         for $j \in [N_3]$ do
20:             $(c_0^{(i,j)}, \cdots, c_{2m-2}^{(i,j)}) \leftarrow$
                    $(p_{\mathbf{C}}^{(i,j)}(\rho_{r_1}^{(P)}), \cdots, p_{\mathbf{C}}^{(i,j)}(\rho_{r_{2m-1}}^{(P)}))\mathbf{G}_{\text{inv}}$
21:             $(p_{\mathbf{C}}^{(i,j)}(\rho_1^{(m)}), \cdots, p_{\mathbf{C}}^{(i,j)}(\rho_m^{(m)})) \leftarrow$
                    $(c_0^{(i,j)}, \cdots, c_{2m-2}^{(i,j)})\tilde{\mathbf{G}}^{(2m-1,m)}(\boldsymbol{\rho}^{(m)})$
22:             $\hat{C}(i,j) \leftarrow$
                    $\frac{2}{m}(p_{\mathbf{C}}^{(i,j)}(\rho_1^{(m)}), \cdots, p_{\mathbf{C}}^{(i,j)}(\rho_m^{(m)}))(1, \cdots, 1)^T$ ▷ $a_i$'s are all $2/m$
23:         end for
24:     end for
25:     return $\hat{\mathbf{C}}$
26: end procedure
```

workers, the partitioning factors $m = 30, 29, 28$ correspond to $s = 1, 3, 5$, respectively, and for the system with 100 workers, the partitioning factors $m = 50, 49$ correspond to $s = 1, 3$, respectively.

Fig. 8 compares the maximum/average condition number of the $(2m-1) \times (2m-1)$ sub-matrices of three decoding matrices $\mathbf{G}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})$ (i.e., Construction 2), $\mathbf{G}^{(2m-1,P)}(\mathbf{u}^{(P)})$, where $\mathbf{u}$ is a P-length vector of equi-spaced points from -1 to 1 (i.e., Construction 2 but using the uniformly spaced points in $\mathbf{u}^{(P)}$ as evaluation points instead of $\boldsymbol{\rho}^{(P)}$) and $\mathbf{M}$ (i.e., MatDot Codes) in two cases, the first where $(x_1, \cdots, x_P) = \boldsymbol{\rho}^{(P)}$ and the other where $(x_1, \cdots, x_P) = \mathbf{u}^{(P)}$. The figure shows that while MatDot Codes provide a reasonable condition number ($\sim 10^{10}$) to distributed systems with sizes up to only 25 worker nodes, Construction 2 with Chebyshev points as evaluation points can afford distributed systems with sizes up to 150 worker nodes for the same condition number bound $\sim 10^{10}$. Another observation from Fig. 8, that shows the effect of choosing the evaluation points on the condition number, is comparing the condition number of Construction 2 with two different choices of evaluation points. Specifically, Fig. 8 shows that, for the same construction, i.e., Construction 2, choosing the evaluation points as the entries of $\boldsymbol{\rho}^{(P)}$ yields well-conditioned matrices compared to choosing the entries of $\mathbf{u}$ as evaluation points, even with the fact that these polynomial roots are computed. The reason behind this is that the computation of the Chebyshev polynomials' roots does not involve any polynomial solving procedures; however, they are computed using the closed form expression in (7).

As a reflection to the significant higher stability of Chebyshev polynomials based OrthoMatDot compared to MatDot Codes, Fig. 9 shows that Chebyshev polynomials based OrthoMatDot provides much more accurate outputs compared to MatDot Codes. For the experiments whose results are shown in Fig. 9, the entries of the input matrices $\mathbf{A}, \mathbf{B}$ are chosen independently according to the standard Gaussian distribution $\mathcal{N}(0,1)$. In addition, for any two input matrices $\mathbf{A}, \mathbf{B}$, let $\hat{\mathbf{C}}$ be the output of the distributed system (which is not necessarily equal to the correct answer $\mathbf{AB}$), we define the relative error between $\mathbf{AB}$ and $\hat{\mathbf{C}}$ to be

$$E_r(\mathbf{AB}, \hat{\mathbf{C}}) = \frac{||\mathbf{AB} - \hat{\mathbf{C}}||_F}{||\mathbf{AB}||_F}. \tag{17}$$

Fig. 9 shows how the maximum relative error (the worst case relative error given a fixed number of parity workers s among all the $P - s$ successful nodes scenarios) grows with the size of the distributed system. In Fig. 9, we plot the average result

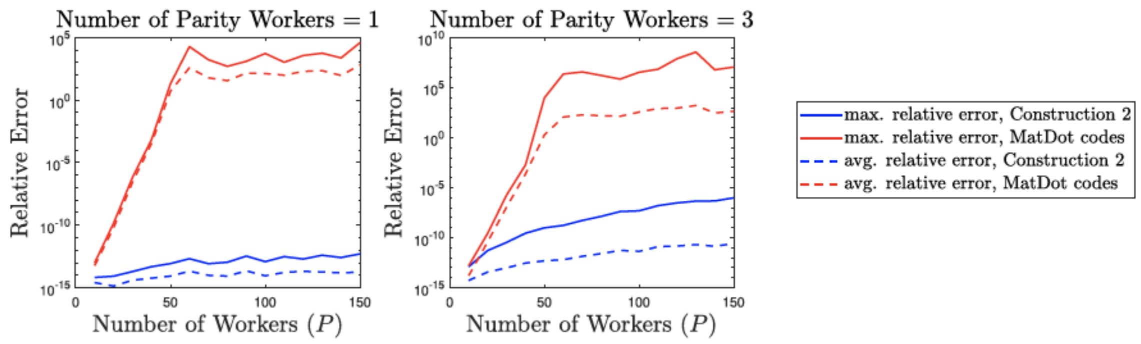


Fig. 9. The growth of the relative error, for both Chebyshev polynomials based OrthoMatDot and MatDot Codes, both using Chebyshev points, with the system size given a fixed number of redundant worker nodes.

of five different realizations of the system at each system size P. The figure shows that MatDot Codes crushes after the size of the system exceeds 50 workers, providing a relative error of around 10^5. On the other hand, our OrthoMatDot construction can support systems with sizes up to 150 worker nodes only allowing for a relative error $< 10^{-5}$. It is also worth mentioning that in our experiments, we use the MATLAB command $inv()$ [31] for matrix inversion. We have also tried matrices inversion through the Bjorck-Pereyra algorithm [33], however, its results were much less accurate than $inv()$, especially for large systems with a number of worker nodes > 50. The fact that the Bjorck-Pereyra algorithm is not accurate, in our case, was already observed in [34]. The reason is that the Bjorck-Pereyra algorithm is essentially based on Newton's formula for polynomial interpolation, and hence is not applicable, since we compare here with a set of powers (i.e., the monomial basis $1, x, x^2, \cdots$) which are not"in line".

VI. OrthoPoly: Low Communication/Computation Numerically Stable Codes for Distributed Marix Multiplication

While MatDot Codes [3] have an optimal recovery threshold of $2m - 1$, they have relatively higher computation cost per worker and worker node to fusion node communication cost as compared to Polynomial Codes [6]. In this section, motivated by the condition number bound in Theorem 5.1, we use the idea of using Chebyshev polynomials to provide a numerically stable code construction for matrix multiplication that has the same low communication/computation costs as Polynomial Codes, as well as the same recovery threshold. However, as will be shown in this section, our proposed codes, denoted by *OrthoPoly*, provides lower numerical errors than Polynomial Codes. In this section, we follow the same system model as in Section IV-A1, and solve the problem statement formulated in Section VI-A. We provide a motivating example in Section VI-B, then we provide the general code construction in Section VI-C. For a fair comparison, we compare our OrthoPoly Codes with the current codes with the same input partitioning, i.e., Polynomial Codes [6], through numerical experiments in Section VI-D. We show through the numerical experiments that our proposed codes achieve lower numerical errors as compared to Polynomial Codes.

A. Problem Formulation

The master node possesses two real-valued input matrices $\mathbf{A}$, $\mathbf{B}$ with dimensions $N_1 \times N_2$, $N_2 \times N_3$, respectively. Every worker node receives from the master node an encoded matrix of $\mathbf{A}$ of dimension $N_1/m \times N_2$ and an encoded matrix of $\mathbf{B}$ of dimension $N_2 \times N_3/n$, and performs matrix multiplication of these two received inputs. Upon performing the matrix multiplication, each worker node sends the result to the fusion node. The fusion node needs to recover the matrix multiplication $\mathbf{AB}$ once it receives the results of any mn worker nodes.

B. Example ($m = n = 3$)

Consider computing the matrix multiplication $\mathbf{AB}$, for some two real matrices $\mathbf{A}, \mathbf{B}$ of dimensions $N_1 \times N_2$ and $N_2 \times N_3$, respectively, over a distributed system of $P \geq 9$ workers such that:

1) Each worker receives an encoded matrix of $\mathbf{A}$ of dimension $N_1/3 \times N_2$, and an encoded matrix of $\mathbf{B}$ of dimension $N_2 \times N_3/3$.
2) The product $\mathbf{AB}$ can be recovered by the fusion node given the results of any 9 worker nodes.

A solution can be as follows: First, matrices $\mathbf{A}, \mathbf{B}$ can be partitioned as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} \mathbf{B}_0 & \mathbf{B}_1 & \mathbf{B}_2 \end{pmatrix}, \tag{18}$$

where, for any $i \in \{0, 1, 2\}$, $\mathbf{A}_i$ has dimension $N_1/3 \times N_2$, and $\mathbf{B}_i$ has dimension $N_2 \times N_3/3$. Next, let

$$p_{\mathbf{A}}(x) = \mathbf{A}_0 T_0(x) + \mathbf{A}_1 T_1(x) + \mathbf{A}_2 T_2(x),$$
$$p_{\mathbf{B}}(x) = \mathbf{B}_0 T_0(x) + \mathbf{B}_1 T_3(x) + \mathbf{B}_2 T_6(x).$$

Now, $p_{\mathbf{A}}(x) p_{\mathbf{B}}(x)$ can be written as

$$\begin{aligned} p_{\mathbf{A}}(x) p_{\mathbf{B}}(x) &= \big(\mathbf{A}_0 T_0(x) + \mathbf{A}_1 T_1(x) + \mathbf{A}_2 T_2(x)\big) \times \\ &\quad (\mathbf{B}_0 T_0(x) + \mathbf{B}_1 T_3(x) + \mathbf{B}_2 T_6(x)) \\ &= \mathbf{A}_0 \mathbf{B}_0 + (\mathbf{A}_1 \mathbf{B}_0 + \frac{1}{2} \mathbf{A}_2 \mathbf{B}_1) T_1(x) + \\ &\quad (\mathbf{A}_2 \mathbf{B}_0 + \frac{1}{2} \mathbf{A}_1 \mathbf{B}_1) T_2(x) + \mathbf{A}_0 \mathbf{B}_1 \, T_3(x) \end{aligned}$$

$$+\frac{1}{2}(\mathbf{A}_1\mathbf{B}_1+\mathbf{A}_2\mathbf{B}_2)T_4(x) + \frac{1}{2}(\mathbf{A}_1\mathbf{B}_2+\mathbf{A}_2\mathbf{B}_1)T_5(x)+\mathbf{A}_0\mathbf{B}_2\ T_6(x) + \frac{1}{2}\mathbf{A}_1\mathbf{B}_2T_7(x)+\frac{1}{2}\mathbf{A}_2\mathbf{B}_2T_8(x). \tag{19}$$

Since $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ is a degree 8 polynomial, once the fusion node receives the output of any 9 workers, it can interpolate $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$, i.e., obtain its matrix coefficients, let such matrix coefficients be $\mathbf{C}_{T_0},\cdots,\mathbf{C}_{T_8}$. Specifically, for any $i\in\{0,\cdots,8\}$, let $\mathbf{C}_{T_i}$ be the matrix coefficient of T_i in $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$. Now, recalling (18), the product $\mathbf{AB}$ can be written as

$$\mathbf{AB}=\begin{pmatrix}\mathbf{A}_0\mathbf{B}_0 & \mathbf{A}_0\mathbf{B}_1 & \mathbf{A}_0\mathbf{B}_2\\ \mathbf{A}_1\mathbf{B}_0 & \mathbf{A}_1\mathbf{B}_1 & \mathbf{A}_1\mathbf{B}_2\\ \mathbf{A}_2\mathbf{B}_0 & \mathbf{A}_2\mathbf{B}_1 & \mathbf{A}_2\mathbf{B}_2\end{pmatrix}. \tag{20}$$

While the obtained set of matrix coefficients $\{\mathbf{C}_{T_i}: i\in\{0,\cdots,8\}\}$ is not equal to $\{\mathbf{A}_i\mathbf{B}_j: i,j\in\{0,1,2\}\}$, $\mathbf{C}_{T_i}$'s are linear combinations of $\mathbf{A}_i\mathbf{B}_j$'s. Specifically, for any $\mathbf{C}_{T_i}$, $i\in\{0,\cdots,8\}$, let $\mathbf{C}_{T_i}^{(k,l)}$ be its (k,l)-th entry, and, for any $i,j\in\{0,1,2\}$, let $(\mathbf{A}_i\mathbf{B}_j)^{(k,l)}$ be the (k,l)-th entry of the product $\mathbf{A}_i\mathbf{B}_j$. Also, let

$$\mathbf{D}=\begin{pmatrix}1&0&0&0&0&0&0&0&0\\0&1&0&0&0&1/2&0&0&0\\0&0&1&0&1/2&0&0&0&0\\0&0&0&1&0&0&0&0&0\\0&0&0&0&1/2&0&0&0&1/2\\0&0&0&0&0&1/2&0&1/2&0\\0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1/2&0\\0&0&0&0&0&0&0&0&1/2\end{pmatrix}. \tag{21}$$

We can write

$$\begin{pmatrix}\mathbf{C}_{T_0}^{(k,l)}\\ \mathbf{C}_{T_1}^{(k,l)}\\ \mathbf{C}_{T_2}^{(k,l)}\\ \mathbf{C}_{T_3}^{(k,l)}\\ \mathbf{C}_{T_4}^{(k,l)}\\ \mathbf{C}_{T_5}^{(k,l)}\\ \mathbf{C}_{T_6}^{(k,l)}\\ \mathbf{C}_{T_7}^{(k,l)}\\ \mathbf{C}_{T_8}^{(k,l)}\end{pmatrix}=\mathbf{D}\begin{pmatrix}(\mathbf{A}_0\mathbf{B}_0)^{(k,l)}\\(\mathbf{A}_1\mathbf{B}_0)^{(k,l)}\\(\mathbf{A}_2\mathbf{B}_0)^{(k,l)}\\(\mathbf{A}_0\mathbf{B}_1)^{(k,l)}\\(\mathbf{A}_1\mathbf{B}_1)^{(k,l)}\\(\mathbf{A}_2\mathbf{B}_1)^{(k,l)}\\(\mathbf{A}_0\mathbf{B}_2)^{(k,l)}\\(\mathbf{A}_1\mathbf{B}_2)^{(k,l)}\\(\mathbf{A}_2\mathbf{B}_2)^{(k,l)}\end{pmatrix}, \tag{22}$$

for any $(k,l)\in[N_1/3]\times[N_3/3]$. Thus, the products $\mathbf{A}_i\mathbf{B}_j, i,j\in\{0,1,2\}$ can be obtained by computing

$$\begin{pmatrix}(\mathbf{A}_0\mathbf{B}_0)^{(k,l)}\\(\mathbf{A}_1\mathbf{B}_0)^{(k,l)}\\(\mathbf{A}_2\mathbf{B}_0)^{(k,l)}\\(\mathbf{A}_0\mathbf{B}_1)^{(k,l)}\\(\mathbf{A}_1\mathbf{B}_1)^{(k,l)}\\(\mathbf{A}_2\mathbf{B}_1)^{(k,l)}\\(\mathbf{A}_0\mathbf{B}_2)^{(k,l)}\\(\mathbf{A}_1\mathbf{B}_2)^{(k,l)}\\(\mathbf{A}_2\mathbf{B}_2)^{(k,l)}\end{pmatrix}=\mathbf{D}^{-1}\begin{pmatrix}\mathbf{C}_{T_0}^{(k,l)}\\ \mathbf{C}_{T_1}^{(k,l)}\\ \mathbf{C}_{T_2}^{(k,l)}\\ \mathbf{C}_{T_3}^{(k,l)}\\ \mathbf{C}_{T_4}^{(k,l)}\\ \mathbf{C}_{T_5}^{(k,l)}\\ \mathbf{C}_{T_6}^{(k,l)}\\ \mathbf{C}_{T_7}^{(k,l)}\\ \mathbf{C}_{T_8}^{(k,l)}\end{pmatrix}, \tag{23}$$

for all $(k,l)\in[N_1/3]\times[N_3/3]$. In the following, we provide the general code construction.

C. OrthoPoly Code Construction

We assume that matrix $\mathbf{A}$ is split horizontally into m equal sub-matrices, of dimension $N_1/m\times N_2$ each, and matrix $\mathbf{B}$ is split vertically into n equal sub-matrices, of dimension $N_2\times N_3/n$ each, as follows:

$$\mathbf{A}=\begin{pmatrix}\mathbf{A}_0\\ \mathbf{A}_1\\ \vdots\\ \mathbf{A}_{m-1}\end{pmatrix},\quad \mathbf{B}=(\mathbf{B}_0\ \mathbf{B}_1\ \ldots\ \mathbf{B}_{n-1}), \tag{24}$$

and define two encoding polynomials $p_{\mathbf{A}}(x)=\sum_{i=0}^{m-1}\mathbf{A}_iT_i(x)$ and $p_{\mathbf{B}}(x)=\sum_{i=0}^{n-1}\mathbf{B}_iT_{im}(x)$, and let $p_{\mathbf{C}}(x)=p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$. We describe, next, the idea of the general code construction. First, for all $r\in[P]$, the master node sends to the r-th worker evaluations of $p_{\mathbf{A}}(x)$ and $p_{\mathbf{B}}(x)$ at $x=\rho_r^{(P)}$, that is, it sends $p_{\mathbf{A}}(\rho_r^{(P)})$ and $p_{\mathbf{B}}(\rho_r^{(P)})$ to the r-th worker. Next, for every $r\in[P]$, the r-th worker node computes the matrix product $p_{\mathbf{C}}(\rho_r^{(P)})=p_{\mathbf{A}}(\rho_r^{(P)})p_{\mathbf{B}}(\rho_r^{(P)})$ and sends the result to the fusion node. Once the fusion node receives the output of any mn worker nodes, it interpolates $p_{\mathbf{C}}(x)$ with respect to the Chebyshev basis $\{T_i\}_{i\geq 0}$. Next, the fusion node recovers the products $\mathbf{A}_i\mathbf{B}_j, i\in\{0,\cdots,m-1\}, j\in\{0,\cdots,n-1\}$, from the matrix coefficients of $p_{\mathbf{C}}(x)$ using a low complexity matrix-vector multiplication, specified in Construction 3. We formally present our **OrthoPoly** Codes in Construction 3. In the following, we explain the notation used in Construction 3: The output of the system is defined as the $N_1\times N_3$ matrix $\hat{\mathbf{C}}$, where the (k,l)-th block of $\hat{\mathbf{C}}$ is the $N_1/m\times N_3/n$ matrix $\hat{\mathbf{C}}_{k,l}$, and the (i,j)-th entry of any matrix $\hat{\mathbf{C}}_{k,l}$ is $\hat{c}_{k,l}^{(i,j)}$. The (i,j)-th entry of the matrix polynomial $p_{\mathbf{C}}(x)$ is denoted as $p_{\mathbf{C}}^{(i,j)}(x)$, and Section III-C defines matrices $\mathbf{G}^{(mn,P)}(\rho^{(P)})$ and $\mathbf{G}_{\mathcal{R}}^{(mn,P)}(\rho^{(P)})$, for any subset $\mathcal{R}=\{r_1,\cdots,r_{mn}\}\subset[P]$. In addition, $\mathbf{H}$ is an $mn\times mn$ matrix of the following form $\mathbf{H}=\begin{pmatrix}\mathbf{H}_0 & \mathbf{H}_1 & \cdots & \mathbf{H}_{n-1}\end{pmatrix}$, where $\mathbf{H}_0$ is an $mn\times m$ matrix with ones on the main diagonal and zeros elsewhere, and for any $i\in\{1,\cdots,n-1\}$,

Construction 3 OrthoPoly: **Inputs:** $\mathbf{A}, \mathbf{B}$, **Output:** $\hat{\mathbf{C}}$

```
1: procedure MasterNode($\mathbf{A}, \mathbf{B}$)    ▷ The master node's procedure
2:     $r \leftarrow 1$
3:     while $r \neq P+1$ do
4:         $p_{\mathbf{A}}(\rho_r^{(P)}) \leftarrow \sum_{i=0}^{m-1} \mathbf{A}_i T_i(\rho_r^{(P)})$
5:         $p_{\mathbf{B}}(\rho_r^{(P)}) \leftarrow \sum_{i=0}^{n-1} \mathbf{B}_i T_{im}(\rho_r^{(P)})$
6:         send $p_{\mathbf{A}}(\rho_r^{(P)}), p_{\mathbf{B}}(\rho_r^{(P)})$ to worker node $r$
7:         $r \leftarrow r+1$
8:     end while
9: end procedure
10:
11: procedure WorkerNode($p_{\mathbf{A}}(\rho_r^{(P)}), p_{\mathbf{B}}(\rho_r^{(P)})$)    ▷ The procedure of worker node $r$
12:     $p_{\mathbf{C}}(\rho_r^{(P)}) \leftarrow p_{\mathbf{A}}(\rho_r^{(P)}) p_{\mathbf{B}}(\rho_r^{(P)})$
13:     send $p_{\mathbf{C}}(\rho_r^{(P)})$ to the fusion node
14: end procedure
15:
16: procedure FusionNode($\{p_{\mathbf{C}}(\rho_{r_1}^{(P)}), \cdots, p_{\mathbf{C}}(\rho_{r_{mn}}^{(P)})\}$)    ▷ The fusion node's procedure, $r_i$'s are distinct
17:     $\mathbf{G}_{\text{inv}} \leftarrow \left(\mathbf{G}_{\mathcal{R}}^{(mn,P)}\right)^{-1}$
18:     for $i \in [N_1/m]$ do
19:         for $j \in [N_3/n]$ do
20:             $(c_0^{(i,j)}, \cdots, c_{mn-1}^{(i,j)}) \leftarrow (p_{\mathbf{C}}^{(i,j)}(\rho_{r_1}^{(P)}), \cdots, p_{\mathbf{C}}^{(i,j)}(\rho_{r_{mn}}^{(P)}))\mathbf{G}_{\text{inv}}$
21:             $(\hat{c}_{0,0}^{(i,j)} \cdots \hat{c}_{m-1,0}^{(i,j)} \cdots\cdots \hat{c}_{0,n-1}^{(i,j)} \cdots \hat{c}_{m-1,n-1}^{(i,j)}) \leftarrow (c_0^{(i,j)}, \cdots, c_{mn-1}^{(i,j)})(\mathbf{H}^{-1})^T$
22:         end for
23:     end for
24:     return $\hat{\mathbf{C}}$
25: end procedure
```

$\mathbf{H}_i$ is an $mn \times m$ matrix of the following structure

$$\mathbf{H}_i = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \iddots & 1/2 \\ 0 & 0 & 0 & \iddots & 0 \\ 0 & 0 & 1/2 & \iddots & \vdots \\ 0 & 1/2 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1/2 & 0 & \cdots & 0 \\ 0 & 0 & 1/2 & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1/2 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix},$$

where the value 1 in the first column is at the $(im+1)$-th row of $\mathbf{H}_i$.

1) Complexity Analyses of OrthoPoly:

Encoding Complexity: Encoding for each worker requires performing two additions, the first one adds m scaled matrices of size $N_1\ N_2/m$ and the other adds n scaled matrices of size $N_2\ N_3/n$, for an overall encoding complexity for each worker of $O(N_1\ N_2 + N_2 N_3)$. Therefore, the overall computational complexity of encoding for P workers is $O(N_1\ N_2 P + N_2 N_3 P)$.

Computational Cost per Worker: Each worker multiplies two matrices of dimensions $N_1/m \times N_2$ and $N_2 \times N_3/n$, requiring $O(N_1 N_2 N_3/mn)$ operations.

Decoding Complexity: Since $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ has degree $mn-1$, the interpolation of $p_{\mathbf{C}}(x)$ requires the inversion of a $mn \times mn$ matrix, with complexity $O(m^3 n^3)$, and performing $N_1 N_3/mn$ matrix-vector multiplications, each of them is between the inverted matrix and a column vector of length mn of the received evaluations of the matrix polynomial $p_{\mathbf{C}}(x)$ at some position $(i,j) \in [N_1/m] \times [N_3/n]$, with complexity $O(N_1 N_3 m^2 n^2/(mn)) = O(N_1 N_3 mn)$. Thus, assuming that $mn \ll N_1, N_3$, the overall decoding complexity is $O(N_1 N_3\ mn)$.

Communication Cost: The master node sends $O(N_1 N_2 P/m + N_2 N_3 P/n)$ symbols, and the fusion node receives $O(N_1\ N_3)$ symbols from the successful worker nodes.

Remark 6.1: With the reasonable assumption that the dimensions of the input matrices $\mathbf{A}, \mathbf{B}$ are large enough such that $N_1, N_2, N_3 \gg m, n, P$, we can conclude that the encoding and decoding costs at the master and fusion nodes, respectively, are negligible compared to the computation cost at each worker node.

D. Numerical Results

Our experiments were performed on the computation environment described in Section III-B, and the codes used in our experiments can be obtained from [32]. In our experiments, the entries of the input matrices $\mathbf{A}, \mathbf{B}$ are chosen independently according to the standard Gaussian distribution $\mathcal{N}(0,1)$. In addition, for any two input matrices $\mathbf{A}, \mathbf{B}$, let $\hat{\mathbf{C}}$ be the output of the distributed system, we define the relative error between $\mathbf{AB}$ and $\hat{\mathbf{C}}$ to be

$$E_r(\mathbf{AB}, \hat{\mathbf{C}}) = \frac{||\mathbf{AB} - \hat{\mathbf{C}}||_F}{||\mathbf{AB}||_F}.$$

Fig. 10 shows how the maximum relative error (the worst case relative error given a fixed number of parity workers s among all the $P-s$ successful nodes scenarios) grows with the size of the distributed system for both Construction 3 and Polynomial Codes. In Fig. 10, we plot the average result of five different realizations of the system at each system size P. The figure shows that Polynomial Codes have unacceptable relative errors after the size of the system exceeds 50 workers, providing a relative error of around 10^5. On the other hand, OrthoPoly can support systems with sizes up to 170 worker nodes only allowing for a relative error $< 10^{-5}$.

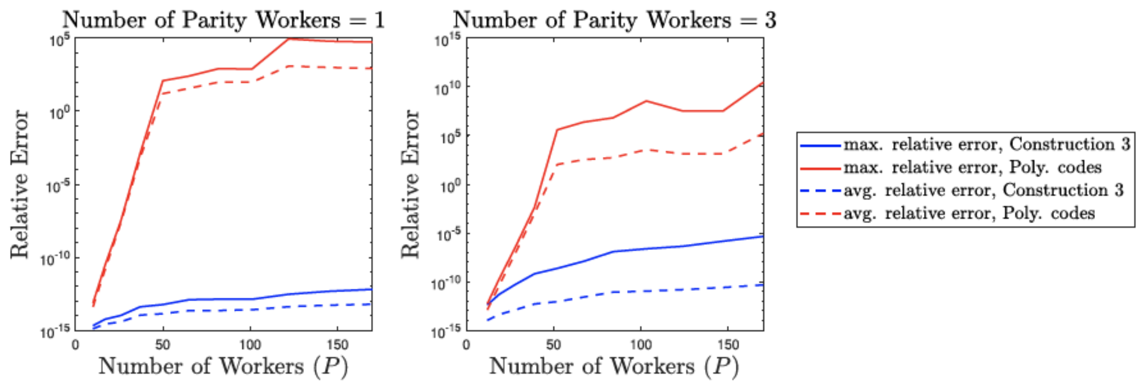


Fig. 10. The growth of the relative error, for both OrthoPoly and Polynomial Codes, both using Chebyshev points, with the system size given a fixed number of redundant worker nodes.

VII. Generalized OrthoMatDot: Numerically Stable Codes for Matrix Multiplication with Communication/Computation-Recovery Threshold Trade-off

Although MatDot Codes [3] have a low recovery threshold of $2m-1$ as compared with Polynomial Codes [6] which have a recovery threshold of mn, MatDot Codes' worker to fusion nodes communication cost and computation cost per worker are higher than Polynomial Codes. Codes proposed in [4], [5], [26], [27] offer a trade-off between the communication/computation cost and the recovery threshold. In this section, we offer an alternative, numerically stable code construction, denoted by *Generalized OrthoMatDot*, that offers a trade-off between communication/computation costs and recovery threshold. We provide in Section VII-A the formal problem statement considered in this section. We describe an example of our construction in Section VII-B, provide the general code construction in Section VII-C, conduct a full forward error analysis for the computation system under a specification of the general code construction in Section VII-D, and describe our numerical experiments in Section VII-E.

A note on the recovery threshold achievable by Generalized OrthoMatDot: Generalized PolyDot Codes [5] and the Entangled Polynomial Codes [26] are monomial based codes that achieve similar communication/computation-recovery threshold trade-offs. Specifically, they achieve a recovery threshold of $m_1m_2m_3+m_2-1$. The Generalized OrthoMatDot Codes we provide here have a recovery threshold of $4m_1m_2m_3-2(m_1m_2+m_2m_3+m_1m_3)+m_1+2m_2+m_3-1$ and are significantly more stable. Note that the recovery threshold of our construction is higher by a factor of at most 4 as compared to the monomials-based codes of [5], [26], for the same communication/computation cost. This increased recovery threshold is due to the fact that Generalized OrthoMatDot Codes are based on Chebyshev polynomials which have the following property: For any $i,j\in\mathbb{N}$, $T_i(x)T_j(x)=1/2\ (T_{i+j}(x)+T_{|i-j|}(x))$. This property allows for a higher number of *undesired* terms in the multiplication of the encoding polynomials $p_{\mathbf{A}}(x),p_{\mathbf{B}}(x)$. In order to avoid combining undesired and desired terms at the same degree, higher degree Chebyshev polynomials have to be used in $p_{\mathbf{B}}(x)$, yielding a higher recovery threshold. It is still an open question whether the recovery thresholds in [5], [26] can be achieved using orthonormal polynomials.

Notably, [26] develops an improved version of the Entangled Polynomial Code that can achieve a smaller recovery threshold by possibly using polynomials expanded/evaluated using non-monomial basis. Specifically, *the Improved Entangled Polynomial Codes* develop a recovery threshold based on the matrix multiplication's bilinear complexity, and our code constructions can have a larger gap in terms of recovery threshold with respect to these code constructions.

To summarize, Table II compares the recovery threshold of our proposed Generalized OrthoMatDot Codes to the Generalized PolyDot Codes [5], the Entangled Polynomial Codes [26], and the Improved Entangled Polynomial Codes [26], where $R(m_1,m_2,m_3)$ is the bilinear complexity of multiplying two $m_1\times m_2$ matrix and $m_2\times m_3$ matrices [26].

A. System Model and Problem Formulation

We consider the same system model and problem formulation as in Section IV-A with the following change: We assume that the master node is allowed to send an encoded $1/m$ fraction of matrix $\mathbf{A}$, and an encoded $1/n$ fraction of matrix $\mathbf{B}$, where m and n are not necessarily equal, and $\mathbf{A}$ and $\mathbf{B}$ are split as follows

$$\mathbf{A}=\begin{pmatrix}\mathbf{A}_{0,0} & \cdots & \mathbf{A}_{0,m_2-1}\\ \vdots & \ddots & \vdots\\ \mathbf{A}_{m_1-1,0} & \cdots & \mathbf{A}_{m_1-1,m_2-1}\end{pmatrix},$$
$$\mathbf{B}=\begin{pmatrix}\mathbf{B}_{0,0} & \cdots & \mathbf{B}_{0,m_3-1}\\ \vdots & \ddots & \vdots\\ \mathbf{B}_{m_2-1,0} & \cdots & \mathbf{B}_{m_2-1,m_3-1}\end{pmatrix}, \tag{25}$$

where m_1,m_2,m_3 divide N_1,N_2,N_3, respectively, and $m=m_1m_2, n=m_2m_3$. In addition, we assume that each worker node receives a linear combination of sub-matrices $\mathbf{A}_{i,j}$, and another linear combination of sub-matrices $\mathbf{B}_{i,j}$.

B. Example ($m_1=m_2=m_3=2$)

Consider computing the matrix multiplication $\mathbf{AB}$, for some two real matrices $\mathbf{A},\mathbf{B}$ of dimensions $N_1\times N_2$ and $N_2\times N_3$,

TABLE II

A RECOVERY THRESHOLDS' COMPARISON BETWEEN OUR PROPOSED GENERALIZED ORTHOMATDOT CODES, AND THE MONOMIALS-BASED CODES IN [5], [26], NAMELY, THE GENERALIZED POLYDOT CODES [5] AND THE ENTANGLED POLYNOMIAL CODES [26], RESPECTIVELY, AS WELL AS THE NON-MONOMIALS-BASED IMPROVED ENTANGLED POLYNOMIAL CODES [26].

Monomials-Based Codes		Non-Monomials-Based Codes	
Generalized PolyDot Codes [5]	Entangled Polynomial Codes [26]	Generalized OrthoMatDot Codes [This section]	Improved Entangled Polynomial Codes [26]
$m_1m_2m_3 + m_2 - 1$	$m_1m_2m_3 + m_2 - 1$	$4m_1m_2m_3 - 2(m_1m_2 + m_2m_3 + m_3m_1) + m_1 + 2m_2 + m_3 - 1$	$2R(m_1, m_2, m_3) - 1$

respectively, over a distributed system of $P \geq 15$ workers such that:

1) Each worker receives an encoded matrix of $\mathbf{A}$ of dimension $N_1/2 \times N_2/2$, and an encoded matrix of $\mathbf{B}$ of dimension $N_2/2 \times N_3/2$.
2) The product $\mathbf{AB}$ can be recovered by the fusion node given the results of any 15 worker nodes.

A solution can be as follows: First, matrices $\mathbf{A}, \mathbf{B}$ can be partitioned as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} \\ \mathbf{B}_{1,0} & \mathbf{A}_{1,1} \end{pmatrix}, \tag{26}$$

where, for $i, j \in \{0, 1\}$, $\mathbf{A}_{i,j}$ has dimension $N_1/2 \times N_2/2$, and $\mathbf{B}_{i,j}$ has dimension $N_2/2 \times N_3/2$. Next, let

$$\begin{aligned} p_{\mathbf{A}}(x) =& \mathbf{A}_{0,0}T_1(x) + \frac{1}{2}\mathbf{A}_{0,1}T_0(x) \\ &+ \mathbf{A}_{1,0}T_{\alpha+1}(x) + \mathbf{A}_{1,1}T_\alpha(x), \\ p_{\mathbf{B}}(x) =& \frac{1}{2}\mathbf{B}_{0,0}T_0(x) + \mathbf{B}_{1,0}T_1(x) \\ &+ \mathbf{B}_{0,1}T_\beta(x) + \mathbf{B}_{1,1}T_{\beta+1}(x), \end{aligned}$$

where α, β to be specified next, and define P distinct real numbers $x_1, x_2, \cdots, x_P$ in the range $[-1, 1]$. For each worker node $r \in [P]$, the master node sends $p_{\mathbf{A}}(x_r)p_{\mathbf{B}}(x_r)$.

Now, in order to specify the best values for α, β, we expand the polynomial $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ in the Chebyshev basis, and then point out some observations.

$$\begin{aligned} p_{\mathbf{A}}(x)p_{\mathbf{B}}(x) = & \frac{1}{4}\mathbf{A}_{0,1}\mathbf{B}_{0,0} + \frac{1}{2}\mathbf{A}_{0,0}\mathbf{B}_{0,0}T_1(x) \\ &+ \frac{1}{2}\mathbf{A}_{0,1}\mathbf{B}_{1,0}T_1(x) + \mathbf{A}_{0,0}\mathbf{B}_{1,0}T_1(x)T_1(x) \\ &+ \frac{1}{2}\mathbf{A}_{1,1}\mathbf{B}_{0,0}T_\alpha(x) + \mathbf{A}_{1,1}\mathbf{B}_{1,0}T_1(x)T_\alpha(x) \\ &+ \frac{1}{2}\mathbf{A}_{1,0}\mathbf{B}_{0,0}T_{\alpha+1}(x) \\ &+ \mathbf{A}_{1,0}\mathbf{B}_{1,0}T_1(x)T_{\alpha+1}(x) \\ &+ \frac{1}{2}\mathbf{A}_{0,1}\mathbf{B}_{0,1}T_\beta(x) + \mathbf{A}_{0,0}\mathbf{B}_{0,1}T_1(x)T_\beta(x) \\ &+ \frac{1}{2}\mathbf{A}_{0,1}\mathbf{B}_{1,1}T_{\beta+1}(x) \\ &+ \mathbf{A}_{0,0}\mathbf{B}_{1,1}T_1(x)T_{\beta+1}(x) \\ &+ \mathbf{A}_{0,1}\mathbf{B}_{1,1}T_1(x)T_{\beta+1}(x) \\ &+ \mathbf{A}_{1,0}\mathbf{B}_{0,1}T_{\alpha+1}(x)T_\beta(x) \\ &+ \mathbf{A}_{1,1}\mathbf{B}_{1,1}T_\alpha(x)T_{\beta+1}(x) \\ &+ \mathbf{A}_{1,0}\mathbf{B}_{1,1}T_{\alpha+1}(x)T_{\beta+1}(x). \end{aligned} \tag{27}$$

Using the property of the Chebyshev polynomials that for any $i, j \in \mathbb{N}$, $T_i(x)T_j(x) = 1/2\ (T_{i+j}(x) + T_{|i-j|}(x))$, (27) can be rewritten as

$$\begin{aligned} p_{\mathbf{A}}(x)p_{\mathbf{B}}(x) = & \frac{1}{4}\mathbf{A}_{0,1}\mathbf{B}_{0,0} + \frac{1}{2}\mathbf{A}_{0,0}\mathbf{B}_{1,0} \\ &+ \frac{1}{2}(\mathbf{A}_{0,0}\mathbf{B}_{0,0} + \mathbf{A}_{0,1}\mathbf{B}_{1,0})T_1(x) \\ &+ \frac{1}{2}\mathbf{A}_{0,0}\mathbf{B}_{1,0}T_2(x) + \frac{1}{2}\mathbf{A}_{1,1}\mathbf{B}_{1,0}T_{\alpha-1}(x) \\ &+ \frac{1}{2}(\mathbf{A}_{1,1}\mathbf{B}_{0,0} + \mathbf{A}_{1,0}\mathbf{B}_{1,0})T_\alpha(x) \\ &+ \frac{1}{2}(\mathbf{A}_{1,0}\mathbf{B}_{0,0} + \mathbf{A}_{1,1}\mathbf{B}_{1,0})T_{\alpha+1}(x) \\ &+ \frac{1}{2}\mathbf{A}_{1,0}\mathbf{B}_{1,0}T_{\alpha+2}(x) + \frac{1}{2}\mathbf{A}_{1,0}\mathbf{B}_{0,1}T_{\beta-\alpha-1}(x) \\ &+ \frac{1}{2}(\mathbf{A}_{1,1}\mathbf{B}_{0,1} + \mathbf{A}_{1,0}\mathbf{B}_{1,1})T_{\beta-\alpha}(x) \\ &+ \frac{1}{2}\mathbf{A}_{1,1}\mathbf{B}_{1,1}T_{\beta-\alpha+1}(x) + \frac{1}{2}\mathbf{A}_{0,0}\mathbf{B}_{0,1}T_{\beta-1}(x) \\ &+ \frac{1}{2}(\mathbf{A}_{0,1}\mathbf{B}_{0,1} + \mathbf{A}_{0,0}\mathbf{B}_{1,1})T_\beta(x) \\ &+ \frac{1}{2}(\mathbf{A}_{0,0}\mathbf{B}_{0,1} + \mathbf{A}_{0,1}\mathbf{B}_{1,1})T_{\beta+1}(x) \\ &+ \frac{1}{2}\mathbf{A}_{0,0}\mathbf{B}_{1,1}T_{\beta+2}(x) + \frac{1}{2}\mathbf{A}_{1,1}\mathbf{B}_{0,1}T_{\beta+\alpha}(x) \\ &+ \frac{1}{2}(\mathbf{A}_{1,0}\mathbf{B}_{0,1} + \mathbf{A}_{1,1}\mathbf{B}_{1,1})T_{\beta+\alpha+1}(x) \\ &+ \frac{1}{2}\mathbf{A}_{1,0}\mathbf{B}_{1,1}T_{\beta+\alpha+2}(x). \end{aligned} \tag{28}$$

Now, note the following regrading $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ in (28):

(i) $\frac{1}{2}(\mathbf{A}_{0,0}\mathbf{B}_{0,0} + \mathbf{A}_{0,1}\mathbf{B}_{1,0})$ is the coefficient of $T_1(x)$,
(ii) $\frac{1}{2}(\mathbf{A}_{1,0}\mathbf{B}_{0,0} + \mathbf{A}_{1,1}\mathbf{B}_{1,0})$ is the coefficient of $T_{\alpha+1}(x)$,
(iii) $\frac{1}{2}(\mathbf{A}_{0,0}\mathbf{B}_{0,1} + \mathbf{A}_{0,1}\mathbf{B}_{1,1})$ is the coefficient of $T_{\beta+1}(x)$,
(iv) $\frac{1}{2}(\mathbf{A}_{1,0}\mathbf{B}_{0,1} + \mathbf{A}_{1,1}\mathbf{B}_{1,1})$ is the coefficient of $T_{\beta+\alpha+1}(x)$.

Since $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ has degree $\beta + \alpha + 2$, and this polynomial is evaluated at distinct value at each worker node, once the fusion node receives the output of any $\beta + \alpha + 3$ worker nodes, it can interpolate $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ and extract the product $\mathbf{AB}$ (i.e., the matrix coefficients of $T_1(x), T_{\alpha+1}(x), T_{\beta+1}(x), T_{\beta+\alpha+1}(x)$). Now, we aim for picking values for α, β such that the degree of $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ is minimal; and hence, the recovery threshold is minimal as well. These minimal values for α, β must be chosen such that the desired coefficients in (i)-(iv) are separate. That is, each of them is neither combined with another desired nor undesired term.

This constraint leads to the following two inequalities:

$$\alpha - 1 > 1, \text{ and } \alpha + 1 < \beta - \alpha - 1,$$

which implies that $\alpha = 3, \beta = 9$. Next, we provide our general code construction for the **Generalized OrthoMatDot Codes**.

C. Generalized OrthoMatDot Code Construction

Theorem 7.1: For the matrix multiplication problem described in Section VII-A computed on the system defined in Section VII-A, there exists a coding strategy with recovery threshold

$$4m_1m_2m_3 - 2(m_1m_2 + m_2m_3 + m_3m_1) + m_1 + 2m_2 + m_3 - 1. \tag{29}$$

Notice that the problem specified in Section VII-A restricts the output matrix of each worker node to be of dimension $N_1/m_1 \times N_3/m_3$, for some positive integers m_1, m_3 that divide N_1, N_3, respectively. This is smaller than the dimensions of the output matrix of each worker node according to the problem specified in Section IV-A2 (i.e., $N_1 \times N_3$) by a factor of m_1m_3. However, according to Theorem 7.1, this **communication advantage, when $m_1 > 1$ or $m_2 > 1$, comes at the expense of a higher recovery threshold compared to OrthoMatDot Codes.**

Remark 7.1 (Notation): For ease of exposition in the remaining of this section, we use $T_0', T_1', T_2', \cdots$ to denote $\frac{1}{2}T_0, T_1, T_2, \cdots$, respectively.

In order to prove Theorem 7.1, we first present a code construction that achieves the recovery threshold in (29), then we prove that the presented code construction is valid. First, note that in the **Generalized OrthoMatDot** code construction, we assume that the two input matrices $\mathbf{A}, \mathbf{B}$ are split as in (25). Also, note that given this partitioning of input matrices, we can write $\mathbf{C} = \mathbf{AB}$, where $\mathbf{C}$ is written as

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_{0,0} & \cdots & \mathbf{C}_{0,m_3-1} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{m_1-1,0} & \cdots & \mathbf{C}_{m_1-1,m_3-1} \end{pmatrix}, \tag{30}$$

and each of $\mathbf{C}_{i,l}$ has dimension $N_1/m_1 \times N_3/m_3$ and can be expressed as $\mathbf{C}_{i,l} = \sum_{j=0}^{m_2-1} \mathbf{A}_{i,j}\mathbf{B}_{j,l}$, for any $i \in \{0,1,\cdots,m_1-1\}$, and $l \in \{0,1,\cdots,m_3-1\}$. Also, let $x_1,\cdots,x_P$ be distinct real numbers in the range $[-1,1]$, and define encoding polynomials

$$p_{\mathbf{A}}(x) = \sum_{i=0}^{m_1-1}\sum_{j=0}^{m_2-1} \mathbf{A}_{i,j}T'_{m_2-1-j+i(2m_2-1)}(x),$$
$$p_{\mathbf{B}}(x) = \sum_{k=0}^{m_2-1}\sum_{l=0}^{m_3-1} \mathbf{B}_{k,l}T'_{k+l(2m_1-1)(2m_2-1)}(x), \tag{31}$$

and let $p_{\mathbf{C}}(x) = p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$. Notice that $p_{\mathbf{C}}(x)$ is a polynomial matrix of degree equals $\deg_{\mathbf{C}} := 4m_1m_2m_3 - 2(m_1m_2 + m_2m_3 + m_3m_1) + m_1 + 2m_2 + m_3 - 2$.

Claim 7.2: For any $i \in \{0,1,\cdots,m_1-1\}$ and $l \in \{0,1,\cdots,m_3-1\}$, $\frac{1}{2}\mathbf{C}_{i,l}$ is the matrix coefficient of $T_{m_2-1+i(2m_2-1)+l(2m_1-1)(2m_2-1)}$ in $p_{\mathbf{C}}(x)$,

The proof of this claim is in Appendix C.

We describe, next, the idea of our proposed **Generalized OrthoMatDot code construction**. First, for all $r \in [P]$, the master node sends to the r-th worker evaluations of $p_{\mathbf{A}}(x)$ and $p_{\mathbf{B}}(x)$ at $x = \rho_r^{(P)}$, that is, it sends $p_{\mathbf{A}}(\rho_r^{(P)})$ and $p_{\mathbf{B}}(\rho_r^{(P)})$ to the r-th worker. Next, for every $r \in [P]$, the r-th worker node computes the matrix product $p_{\mathbf{C}}(\rho_r^{(P)}) = p_{\mathbf{A}}(\rho_r^{(P)})p_{\mathbf{B}}(\rho_r^{(P)})$ and sends the result to the fusion node. Once the fusion node receives the output of any $\deg_{\mathbf{C}}+1$ worker nodes, it interpolates $p_{\mathbf{C}}(x)$ with respect to the **Chebyshev basis** $\{T_i\}_{i\geq 0}$.

We formally present our **Generalized OrthoMatDot** code construction in Construction 4. In the following, we explain the notation used in Construction 4. The output of the system is the $N_1 \times N_3$ matrix $\hat{\mathbf{C}}$, where the (k,l)-th block of $\hat{\mathbf{C}}$ is the $N_1/m_1 \times N_3/m_3$ matrix $\hat{\mathbf{C}}_{k,l}$, and the (i,j)-th entry of any matrix $\hat{\mathbf{C}}_{k,l}$ is $\hat{c}_{k,l}^{(i,j)}$. The (i,j)-th entry of the matrix polynomial $p_{\mathbf{C}}(x)$ is denoted as $p_{\mathbf{C}}^{(i,j)}(x)$, and Section III-C defines matrices $\mathbf{G}^{(\deg_{\mathbf{C}}+1,P)}(\boldsymbol{\rho}^{(P)})$ and $\mathbf{G}_{\mathcal{R}}^{(\deg_{\mathbf{C}}+1,P)}(\boldsymbol{\rho}^{(P)})$, for any subset $\mathcal{R} = \{r_1,\cdots,r_{\deg_{\mathbf{C}}+1}\} \subset [P]$.

Now, we prove Theorem 7.1.

Proof of Theorem 7.1: To prove the theorem, it suffices to prove that Construction 4 is valid. Noting that $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ has degree $4m_1m_2m_3 - 2(m_1m_2 + m_2m_3 + m_3m_1) + m_1 + 2m_2 + m_3 - 2$ and every worker node sends an evaluation of $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ at a distinct point, once the fusion node receives the output of any $4m_1m_2m_3 - 2(m_1m_2 + m_2m_3 + m_3m_1) + m_1 + 2m_2 + m_3 - 1$ worker node, it can interpolate $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ (i.e., obtain all its matrix coefficients). This includes the coefficients of $T_{m_2-1+i(2m_2-1)+l(2m_1-1)(2m_2-1)}$ for all $i \in \{0,1,\cdots,m_1-1\}$, and $l \in \{0,1,\cdots,m_3-1\}$, i.e., $\mathbf{C}_{i,l}$, for all $i \in \{0,1,\cdots,m_1-1\}$, and $l \in \{0,1,\cdots,m_3-1\}$ (Claim 7.2), which completes the proof. □

Next, we provide the different complexity analyses of the **Generalized OrthoMatDot Codes**.

1) Complexity Analyses of Generalized OrthoMatDot:

Encoding Complexity: Encoding for each worker requires performing two additions, the first one adds m_1m_2 scaled matrices of size $N_1\ N_2/(m_1m_2)$ and the other adds $m_2\ m_3$ scaled matrices of size $N_2\ N_3/(m_2m_3)$, for an overall encoding complexity for each worker of $O(N_1\ N_2 + N_2N_3)$. Therefore, the overall computational complexity of encoding for P workers is $O(N_1\ N_2P + N_2N_3P)$.

Computational Cost per Worker: Each worker multiplies two matrices of dimensions $N_1/m_1 \times N_2/m_2$ and $N_2/m_2 \times N_3/m_3$, requiring $O(N_1N_2N_3/(m_1m_2m_3))$ operations.

Decoding Complexity: Since $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ has degree $k - 1 := 4m_1m_2m_3 - 2(m_1m_2 + m_2m_3 + m_3m_1) + m_1 + 2m_2 + m_3 - 2$, the interpolation of $p_{\mathbf{C}}(x)$ requires the inversion of a $k\times k$ matrix, with complexity $O(k^3) = O(m_1^3m_2^3m_3^3)$, and performing $N_1N_3/(m_1m_3)$ matrix-vector multiplications, each of them is between the inverted matrix and a column vector of length k of the received evaluations of the matrix polynomial $p_{\mathbf{C}}(x)$ at some position $(i,j) \in [N_1/m_1] \times [N_3/m_3]$, with complexity $O(N_1N_3k^2/(m_1m_3)) = O(N_1N_3m_1m_2^2m_3)$. Thus, assuming that $m_1, m_3 \ll N_1, N_3$, the overall decoding complexity is $O(N_1N_3m_1m_2^2m_3) = O(N_1N_3\ mn)$.

Construction 4 **Generalized OrthoMatDot: Inputs: $\mathbf{A}, \mathbf{B}$, Output: $\hat{\mathbf{C}}$**

```
1: procedure MasterNode($\mathbf{A}, \mathbf{B}$)    ▷ The master node's procedure
2:     $r \leftarrow 1$
3:     while $r \neq P+1$ do
4:         $p_{\mathbf{A}}(\rho_r^{(P)}) \leftarrow \sum_{i=0}^{m_1-1}\sum_{j=0}^{m_2-1} \mathbf{A}_{i,j} T'_{m_2-1-j+i(2m_2-1)}(\rho_r^{(P)})$
5:         $p_{\mathbf{B}}(\rho_r^{(P)}) \leftarrow \sum_{k=0}^{m_2-1}\sum_{l=0}^{m_3-1} \mathbf{B}_{k,l} T'_{k+l(2m_1-1)(2m_2-1)}(\rho_r^{(P)})$
6:         send $p_{\mathbf{A}}(\rho_r^{(P)}), p_{\mathbf{B}}(\rho_r^{(P)})$ to worker node $r$
7:         $r \leftarrow r+1$
8:     end while
9: end procedure
10:
11: procedure WorkerNode($p_{\mathbf{A}}(\rho_r^{(P)}), p_{\mathbf{B}}(\rho_r^{(P)})$)    ▷ The procedure of worker node $r$
12:     $p_{\mathbf{C}}(\rho_r^{(P)}) \leftarrow p_{\mathbf{A}}(\rho_r^{(P)}) p_{\mathbf{B}}(\rho_r^{(P)})$
13:     send $p_{\mathbf{C}}(\rho_r^{(P)})$ to the fusion node
14: end procedure
15:
16: procedure FusionNode($\{p_{\mathbf{C}}(\rho_{r_1}^{(P)}), \cdots, p_{\mathbf{C}}(\rho_{r_{\deg_{\mathbf{C}}+1}}^{(P)})\}$)
        ▷ The fusion node's procedure, $r_i$'s are distinct
17:     $\mathbf{G}_{\text{inv}} \leftarrow \left(\mathbf{G}_{\mathcal{R}}^{(\deg_{\mathbf{C}}+1,P)}\right)^{-1}$
18:     for $i \in [N_1/m_1]$ do
19:         for $j \in [N_3/m_3]$ do
20:             $(c_0^{(i,j)}, \cdots, c_{\deg_{\mathbf{C}}}^{(i,j)}) \leftarrow (p_{\mathbf{C}}^{(i,j)}(\rho_{r_1}^{(P)}), \cdots, p_{\mathbf{C}}^{(i,j)}(\rho_{r_{\deg_{\mathbf{C}}+1}}^{(P)}))\mathbf{G}_{\text{inv}}$
21:             for $k \in [m_1]$ do
22:                 for $l \in [m_3]$ do
23:                     $\hat{c}_{k,l}^{(i,j)} \leftarrow 2\, c^{(i,j)}_{m_2-1+(k-1)(2m_2-1)+(l-1)(2m_1-1)(2m_2-1)}$
24:                 end for
25:             end for
26:         end for
27:     end for
28:     return $\hat{\mathbf{C}}$
29: end procedure
```

Communication Cost: The master node sends $O(N_1N_2P/(m_1m_2) + N_2N_3P/(m_2m_3))$ symbols, and the fusion node receives $O(N_1\ N_3\ m_2)$ symbols from the successful worker nodes.

Remark 7.2: With the reasonable assumption that the dimensions of the input matrices $\mathbf{A}, \mathbf{B}$ are large enough such that $N_1, N_2, N_3 \gg m_1, m_2, m_3, P$, we can conclude that the encoding and decoding costs at the master and fusion nodes, respectively, are negligible compared to the computation cost at each worker node.

D. Forward Error Analysis

Following the standard approach in numerical analysis, we conduct a forward error analysis for the entire computation system in Appendix D for the Generalized OrthoMatDot Codes. The analysis provides an upper bound on the forward error at each stage of the computation system separately (i.e., encoding/worker computation/decoding). In Theorem D.1, we compose these error analyses to develop an upper bound on the end-to-end error of the overall system, including encoding and decoding, in terms of the machine precision. The result formulated in Theorem D.1 implies that given a machine epsilon[5] of ϵ, and assuming inputs' relative errors of $O(\epsilon)$, and letting the norms of the inputs be $O(1/\epsilon)$, the error is shown to be $O\left(\frac{\kappa\epsilon}{1-\kappa\epsilon}\right)$ where κ is the condition number (note that this is not surprising, e.g., see Theorem 3.1 in [20]). This implies that if $\kappa \ll 1/\epsilon$, then the relative error at the output is $O(\epsilon\kappa)$. Indeed, this formally shows that minimizing the condition number of the decoding matrices is the crucial factor in the numerical accuracy of the output.

E. Numerical Results

Our experiments were performed on the computation environment described in Section III-B, and the codes used in the experiments can be obtained from [32]. In our experiments on Construction 4, we considered distributed systems with $P = 16, 25$ worker nodes. Fig. 11 shows that, for every examined system, the condition number of the interpolation matrix using the Generalized OrthoMatDot Codes is less than its counterpart codes in [5], [26]. The results in Fig. 11 also show that, for the same system, as the partitioning factor m_1 decreases (i.e., as the redundancy in worker nodes increases), the stability of the Generalized OrthoMatDot code construction decreases; however, it is still better than the monomial-basis based codes in [5], [26]. Specifically, due to the direct relation between the partitioning factor m_1 and the maximum number of stragglers s to be tolerated at any fixed total number of workers P, and fixed factors $m_2 = 2, m_3 = 1$, i.e., in this case $s = P - 3m_1$, Fig. 11 shows the relation between the condition number and the maximum number of stragglers to be tolerated at each system, e.g., for the system with 16 workers, the partitioning factors $m_1 = 5, 4, 3$ correspond to $s = 1, 4, 7$, respectively, and for the system with 25 workers, the partitioning factors $m = 8, 7, 6$ correspond to $s = 1, 4, 7$, respectively.

VIII. Numerically Stable Lagrange Coded Computing

In this section, we study the numerical stability of Lagrange coded computing [12] that lifts coded computing beyond matrix-vector and matrix-matrix multiplications, to multivariate polynomial computations. As shown in [12], Lagrange coded computing has applications in gradient coding, privacy and secrecy. Our main contribution here is to develop a numerically stable approach towards Lagrange coded computing inspired by our result of Theorem 5.1. In particular, our contribution involves (a) careful choice of evaluation points, and (b) a careful decoding algorithm that involves inversion of the

[5]The machine epsilon of a floating point system is an upper bound on the relative error due to rounding real numbers to the closest floating point number in the system, e.g., in the IEEE Standard for Floating Point Arithmetic 754 with double precision, the machine epsilon $\epsilon \approx 10^{-16}$.

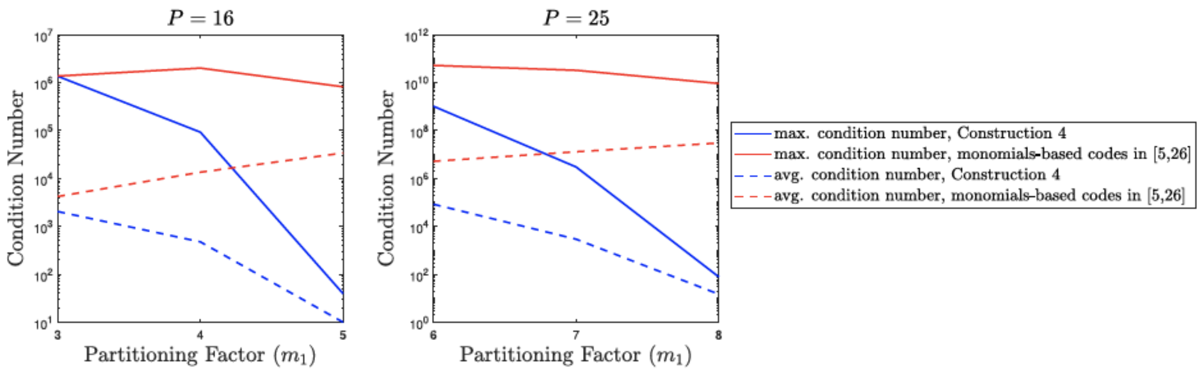


Fig. 11. Comparison between the condition number of the interpolating matrix of the **Generalized OrthoMatDot Codes** and the monomials-based codes in [5], [26] in two distributed systems, one with 16 worker nodes and the other with 25 worker nodes, at different partitioning factors m_1, and fixed $m_2 = 2, m_3 = 1$.

appropriate Chebyshev Vandermonde matrix. We describe the system model in Section VIII-A. We overview the Lagrange coded computing technique of [12] in Section VIII-B. We describe our numerically stable approach in Section VIII-C, and present the results of our numerical experiments in Section VIII-D.

A. System Model and Problem Formulation

We consider, for this section, the distributed computing framework depicted in Fig. 12, that is used in [12] and consists of a master node, P worker nodes, and a fusion node where the only communication allowed is from the master node to the different worker nodes and from the worker nodes to the fusion node. The worker nodes have a prior knowledge of a polynomial function of interest $f : \mathbb{R}^d \to \mathbb{R}^v$ of degree $\deg(f)$, where $d, v \in \mathbb{N}^+$. In addition, the master node possesses a set of data points $\mathcal{X} = \{X_1, \cdots, X_m\}$, where $X_i \in \mathbb{R}^d$, $i \in [m]$. For every worker node $i \in [P]$, the master node is allowed to send some encoded vector $\tilde{X}_i(X_1, \cdots, X_m) \in \mathbb{R}^d$. Once a worker node receives the encoded vector on its input, it evaluates f at this encoded vector and sends the evaluation to the fusion node. That is, for $i \in [P]$, worker node i receives $\tilde{X}_i$ on its input, evaluates $f(\tilde{X}_i)$, then it sends the result to the fusion node. Finally, the fusion node is expected to numerically stably decode the set of evaluations $\mathcal{F} = \{f(X_1), \cdots, f(X_m)\}$ after it receives the output of any K worker nodes.

B. Background on Lagrange Coded Computing

In this section, we review the baseline Lagrange coded computing method introduced in [12] considering the framework in Section VIII-A. Notice that although the method in [12] is more general, here, for simplicity, we limit our discussion to the *systematic* Lagrange coded computing. That is, we assume that for $i \in [m]$, worker node i receives the i-th data point from the master node. In other words, we assume that $\tilde{X}_i = X_i, i \in [m]$. Now, the encoding procedure goes as follows: First, let $x_1, \cdots, x_P$ be distinct real values, an encoding function $g(x)$ is defined as:

$$g(x) = \sum_{i=1}^{m} X_i \prod_{j \in [m]-i} \frac{x - x_j}{x_i - x_j}. \quad (32)$$

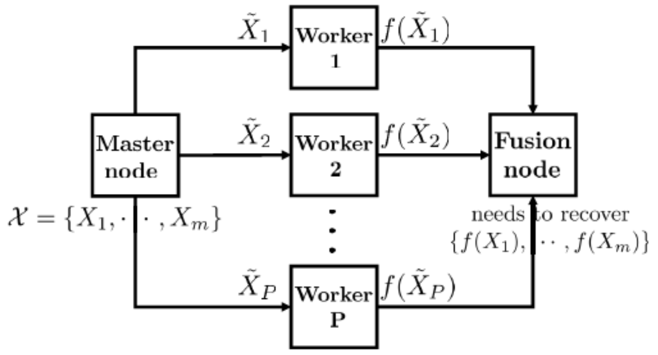


Fig. 12. The Lagrange coded computing system framework.

Given this encoding function, the master node sends the encoded vector $\tilde{X}_i = g(x_i)$ to the worker node i, for every $i \in [P]$. Notice that the encoding function $g(x)$ indeed leads to a systematic encoding since $\tilde{X}_i = g(x_i) = X_i$, for all $i \in [m]$. Every worker node i computes $f(\tilde{X}_i)$ upon the reception of $\tilde{X}_i$, and sends the result to the fusion node. The fusion node waits till receiving the output of any $K := (m-1)\deg(f)+1$. Since $f(g(x))$ has degree $(m-1)\deg(f)$ in x, the fusion node is able to interpolate $f(g(x))$ after receiving the outputs of any $(m-1)\deg(f)+1$, i.e., K, worker nodes. Since $g(x_i) = X_i, i \in [m]$, the fusion nodes evaluates $\{f(g(x_1)), \cdots, f(g(x_m))\}$ to obtain $\{f(X_1), \cdots, f(X_m)\}$.

C. Numerically Stable Lagrange Coded Computing

Lagrange coded computing requires performing an interpolation at the fusion node to recover the polynomial $f(g(x))$. Performing the interpolation by obtaining the coefficients of the polynomial in a monomial basis requires inverting a square Vandermonde matrix which is numerically unstable. Noting that the first ℓ Cheybshev polynomials also forms a basis for degree $\ell - 1$ polynomials, we provide an alternative decoding procedure whose key idea is to find the coefficients of polynomial $f(g(x))$ in the basis of Chebyshev polynomials. Thereby, our decoding procedure involves inverting the Chebyshev-

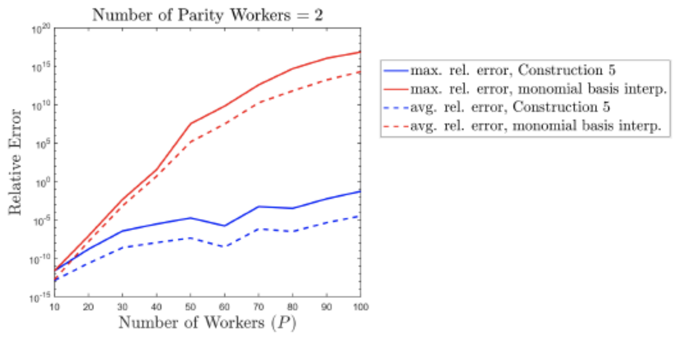


Fig. 13. The growth of the relative error, for Construction 5, using both Chebyshev basis interpolation and monomial basis interpolation, both using Chebyshev points, with the system size given a fixed number of redundant worker nodes equals 2.

Vandermonde matrix[6]. Guided by Theorem 5.1, we choose the evaluation points to be the P-point Chebyshev grid $\rho^{(P)}$ to obtain a decoding procedure that is more stable than one that uses the monomial basis.

Our numerically stable algorithm for Lagrange coded computing is formally described in Construction 5. In the following, we explain the notation used in Construction 5. We let the polynomial at the i-th entry of $f(g(x))$ be denoted $f^{(i)}(x)$ and written as $f^{(i)}(x) = \sum_{l=0}^{K-1} c_l^{(i)} T_l(x)$. Following the notation in Section III-C, we use the Chebyshev-Vandermonde matrices $\mathbf{G}^{(K,P)}(\rho^{(P)})$, and $\mathbf{G}_{\mathcal{R}}^{(K,P)}(\rho^{(P)})$, for any subset $\mathcal{R} = \{r_1, \cdots, r_K\} \subset [P]$, we also define the matrix $\mathbf{G}_{[m]}^{(K,P)}(\rho^{(P)})$. Finally, we assume that our construction returns as output the set of evaluations $\hat{\mathcal{F}} = \{\hat{f}(X_1), \cdots, \hat{f}(X_m)\}$, where for each $\hat{f}(X_i), i \in [m]$, we have $\hat{f}(X_i) = (\hat{f}^{(1)}(x_i), \cdots, \hat{f}^{(v)}(x_i))$, where for every $i \in [m], j \in [v], \hat{f}^{(j)}(x_i)$ and $f^{(j)}(x_i)$ would be the same if the machine had infinite precision.

In the following, we show through numerical experiments the stability of our proposed Construction 5.

Construction 5 Numerically Stable Lagrange Coded Computing **Inputs**: $f, \mathcal{X} = \{X_1, \cdots, X_m\}$, **Output**: $\hat{\mathcal{F}} = \{\hat{f}(X_1), \cdots, \hat{f}(X_m)\}$

```
1: procedure MASTERNODE($\mathcal{X}$)          ▷ The master node's procedure
2:     $r \leftarrow 1$
3:     while $r \neq P+1$ do
4:         if $r \in [m]$ then
5:             $\tilde{X}_r \leftarrow X_i$
6:         else
7:             $\tilde{X}_r \leftarrow \sum_{i=1}^{m} X_i \prod_{j\in[m]-i} \frac{\rho_r^{(P)} - \rho_j^{(P)}}{\rho_i^{(P)} - \rho_j^{(P)}}$
8:         end if
9:         send $\tilde{X}_r$ to worker node $r$
10:        $r \leftarrow r+1$
11:    end while
12: end procedure
13:
14: procedure WORKERNODE($f, \tilde{X}_r$)     ▷ The procedure of worker node $r$
15:     $\mathrm{Out}_r \leftarrow f(\tilde{X}_r)$
16:     send $\mathrm{Out}_r$ to the fusion node
17: end procedure
18:
19: procedure FUSIONNODE($\mathrm{Out}_{r_1}, \cdots, \mathrm{Out}_{r_K}$)▷ The fusion node's procedure, $r_i$'s are distinct
20:     $\mathbf{G}_{\mathrm{inv}} \leftarrow \left(\mathbf{G}_{\mathcal{R}}^{(K,P)}\right)^{-1}$
21:     for $i \in [v]$ do
22:         $(c_0^{(i)}, \cdots, c_{K-1}^{(i)}) \leftarrow (\mathrm{Out}_{r_1}^{(i)}, \cdots, \mathrm{Out}_{r_K}^{(i)})\mathbf{G}_{\mathrm{inv}}$
23:         $(\hat{f}^{(i)}(x_1), \cdots, \hat{f}^{(i)}(x_m)) \leftarrow$
                $(c_0^{(i)}, \cdots, c_{K-1}^{(i)})\mathbf{G}_{[m]}^{(K,P)}$
24:     end for
25:     return $\hat{\mathcal{F}}$
26: end procedure
```

D. Numerical Results

Our experiments were performed on the computation environment described in Section III-B, and the codes used in our experiments can be obtained from [32]. In our experiments, we assume that we have a distributed system of P worker nodes, $m = P - 2$ data points/input vectors $X_1, \cdots, X_m$, each of them is of dimension $d = 10$, where each entry of every input vector is picked independently, according to the standard Gaussian distribution $\mathcal{N}(0,1)$. The function of interest in this system is $f(X) = Y^T X$, where Y is some d-dimensional vector with entries picked independently according to the standard Gaussian distribution $\mathcal{N}(0,1)$. In our experiments, we compare between Construction 5, where the Chebyshev basis is used for interpolation, and the case where the monomial basis is used for interpolation instead. Let $\hat{\mathbf{f}} = (\hat{f}(X_1) \cdots \hat{f}(X_m))$ be the system's output vector, and $\mathbf{f} = (f(X_1) \cdots f(X_m))$ be the correct output vector, we define the relative error between $\mathbf{f}$ and $\hat{\mathbf{f}}$ to be

$$E_r(\mathbf{f}, \hat{\mathbf{f}}) = \frac{||\mathbf{f} - \hat{\mathbf{f}}||_2}{||\mathbf{f}||_2}. \tag{33}$$

The results, shown in Fig. 13, illustrates that using the Chebyshev basis for interpolation provides less relative error/higher stability than the monomial basis at every system size. Fig. 13 also shows that under a certain relative error constraint, Construction 5 provides higher scalability than the monomial basis case. Specifically, let us assume that a relative error up to 0.1 can be tolerated, Fig. 13 shows that the monomial-basis interpolation construction can support systems with a number of worker nodes only less than 40. However, for the same relative error constraint, Construction 5 can support systems with a number of worker nodes up to 100.

Remark 8.1: The Improved Entangled Polynomial Codes [26] are based on Lagrange polynomials and, consequently, avoid the monomial basis. Because these Improved Entangled Polynomial Codes use the Lagrange basis, the evaluation

[6]Since both systematic and non-systematic Lagrange coded computing require the inversion of the same Chebyshev-Vandermonde matrix, our numerically stable decoding procedure in Construction 5 naturally extends to non-systematic Lagrange coded computing, with the only difference is in the last step of evaluating $f(g(x))$ at $x_1, \cdots, x_m$, where in the non-systematic case, $f(g(x))$ is instead evaluated at some predefined values $y_1, \cdots, y_m$ such that $g(y_i) = X_i$ for all $i \in [m]$.

points prescribed by Construction 5 can be applied for an improved numerical stability at the decoding.

IX. Concluding Remarks

In this paper, we developed numerically stable codes for matrix-matrix multiplication and Lagrange coded computing. A distinctive character of our work is the infusion of principles of numerical approximation theory into coded computing towards the end goal of numerical stability. In particular, our work is marked by the use of orthogonal polynomials for encoding, Gauss quadrature techniques for decoding and new bounds on the condition number of Chebyshev Vandermonde matrices. Notably, our constructions obtain the same recovery threshold as MatDot Codes and Polynomial Codes for matrix multiplication as well as for Lagrange Coded Computing. However, our construction in Section VII obtains a weaker (higher) recovery threshold than previous constructions [5], [26] for the problem of coded matrix multiplication when the computation/communication cost is constrained to be lower than that of MatDot Codes. The study of numerically stable codes for this application with the same recovery threshold as [5], [26] remains open.

While our paper focuses on applications where polynomial based encoding are particularly useful, our results might be useful for other applications as well. For instance, for the simple matrix multiplication problem $\mathbf{A}\mathrm{x}$ performed in a distributed setting over P worker nodes, where the goal is to encode $\mathbf{A}$ such that each worker stores a partition $1/m$ of matrix $\mathbf{A}$, it is well known that MDS type codes can be used [13], [28]. Specifically, let $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}$ and let $\mathbf{H} = (h_{ij})$ be an $m \times P$ matrix where every $m \times m$ submatrix of $\mathbf{H}$ has a full rank of m. Then the p-th worker for $p \in \{1, 2, \ldots, P\}$ can compute $\left(\sum_{i=1}^{m} h_{ip}\mathbf{A}_i\right)\mathrm{x}$; the product $\mathbf{A}\mathrm{x}$ can be recovered from any m of the P nodes. The instinctual, Reed-Solomon inspired solution of choosing $\mathbf{H}$ to be a Vandermode matrix is ill-conditioned over real numbers. Note however that, unlike the matrix multiplication problem, the matrix $\mathbf{H}$ does not need to have a polynomial structure. Indeed, choosing $\mathbf{H}$ to be a random Gaussian matrix leads to well-conditioned solutions with high probability. In particular, the following result follows from elementary arguments that build on [35].

Theorem 9.1: Let $\mathbf{H}$ be an $m \times P$ matrix, $P \geq m \geq 3$, and let the entries of $\mathbf{H}$ be independent and identically distributed standard Gaussian random variables. Then,

$$\Pr\left(\kappa_2^{max}(\mathbf{H}) > mP^{2(P-m)}\right) < \frac{5.6}{P^{(P-m)}}.$$

The theorem which is proved in Appendix E, formally demonstrates that for a fixed number of redundant workers $s = P - m$, the worst case condition number grows as $O(mP^{2s})$ with high probability, that is, it is linear in the dimension m. However, the random Gaussian matrix approach has two drawbacks: (i) for a given realization of the random variables, it is difficult to verify whether it is well-conditioned, and (ii) the lack of structure could lead to more complex decoding. Our result of Theorem 5.1 also indicates that choosing $\mathbf{H} = \mathbf{G}^{(m,P)}(\rho^{(P)})$, i.e., to be a Chebyshev Vandermonde matrix, naturally provides a well-conditioned solution to this problem. Another solution for the matrix-vector multiplication problem is provided in [25] via universally decodable matrices [36]; in this work numerical stability is demonstrated empirically.

It is, however, important to note that the problems resolved in our paper here are more restrictive since matrix multiplication codes - where both matrices are to be encoded so that the product can be recovered - require much more structure than matrix multiplication where only one matrix is to be encoded. For instance, random Gaussian encoding does not naturally work for matrix multiplication to get a recovery threshold of $2m - 1$, and it is not clear whether the solution of [25] is applicable either. The utility of Chebyshev-Vandermonde matrices for a variety of coded computing problems including matrix-vector multiplication, matrix multiplication and Lagrange coded computing motivates the study of low-complexity decoding and error correction mechanisms for these systems.

Appendix A
Proof of Claim 4.2

We have,

$$\begin{aligned}
&\int_a^b p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)w(x)dx \\
&= \int_a^b \left(\sum_{i=0}^{m-1} \mathbf{A}_i q_i(x)\right)\left(\sum_{j=0}^{m-1} \mathbf{B}_j q_j(x)\right) w(x)dx \\
&= \int_a^b \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} \mathbf{A}_i\mathbf{B}_j q_i(x)q_j(x)w(x)dx \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} \mathbf{A}_i\mathbf{B}_j \int_a^b q_i(x)q_j(x)w(x)dx \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} \mathbf{A}_i\mathbf{B}_j \ \langle q_i, q_j\rangle \\
&= \sum_{i=0}^{m-1} \mathbf{A}_i\mathbf{B}_i \\
&= \mathbf{AB}. \qquad (34)
\end{aligned}$$

In addition, noting that $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ (i.e., $p_{\mathbf{C}}(x)$) is of degree $2m-2$ (less than $2m$), Theorem 3.2 implies that

$$\begin{aligned}
\int_a^b p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)w(x)dx &= \sum_{r=1}^{m} a_r p_{\mathbf{A}}(\eta_r)p_{\mathbf{B}}(\eta_r) \\
&= \sum_{r=1}^{m} a_r p_{\mathbf{C}}(\eta_r). \qquad (35)
\end{aligned}$$

Finally, combining (34) and (35) completes the proof. □

Appendix B
Proof of Theorem 5.1

We use the following trigonometric identity in our proof.

Lemma B.1: For $n \geq 0$, let x_i be chosen as (7). Then $\prod_{j\neq i}(x_i - x_j) = (-1)^{i-1}\frac{2^{1-n}n}{\sin(\frac{(2i-1)\pi}{2n})}$

Proof: Note that $2^{n-1}\prod_{i=1}^{n}(x - x_i) = T_n(x) = \cos(n\cos^{-1}(x))$. Therefore,

$$2^{n-1}\prod_{j\neq i}(x_i - x_j) = T_n'(x_i) = \frac{n}{\sqrt{1-x_i^2}}\sin(n\cos^{-1}(x_i))$$

where $T_n'(x)$ denotes the derivative of $T_n(x)$. Using $x_i = \cos(\frac{(2i-1)\pi}{2n})$ above we get the desired result. □

Proof of Theorem 5.1: We show that any square submatrix of $\mathbf{G}^{(n-s,n)}(\boldsymbol{\rho}^{(n)})$ formed by any $n-s$ columns of $\mathbf{G}^{(n-s,n)}(\boldsymbol{\rho}^{(n)})$ satisfies the bound stated in the theorem. Let $\mathcal{S}$ be a subset of $[n]$ such that $|\mathcal{S}| = s$, for some $s < n$, and define $\mathbf{G}^{(n-s,n)}_{[n]-\mathcal{S}}(\boldsymbol{\rho}^{(n)})$ to be the square $(n-s)\times(n-s)$ submatrix of $\mathbf{G}^{(n-s,n)}(\boldsymbol{\rho}^{(n)})$ after removing the columns with indices in $\mathcal{S}$. Recalling the structure of $\mathbf{G}^{(n-s,n)}(\boldsymbol{\rho}^{(n)})$ from (13), we can write it as

$$\mathbf{G}^{(n-s,n)}(\boldsymbol{\rho}^{(n)}) = \begin{pmatrix} T_0(\rho_1^{(n)}) & \cdots & T_0(\rho_n^{(n)}) \\ \vdots & \ddots & \vdots \\ T_{n-s-1}(\rho_1^{(n)}) & \cdots & T_{n-s-1}(\rho_n^{(n)}) \end{pmatrix}.$$

Moreover, for any $\mathcal{S} \subset [n]$ such that $|\mathcal{S}| = s$, we can write

$$\mathbf{G}^{(n-s,n)}_{[n]-\mathcal{S}}(\boldsymbol{\rho}^{(n)}) := \mathbf{G}^{(n-s,n)}_{\Gamma} := \begin{pmatrix} T_0(\gamma_1) & \cdots & T_0(\gamma_{n-s}) \\ \vdots & \ddots & \vdots \\ T_{n-s-1}(\gamma_1) & \cdots & T_{n-s-1}(\gamma_{n-s}) \end{pmatrix},$$

where $\Gamma = (\gamma_1, \gamma_2, \cdots, \gamma_{n-s}) = (\rho_{g_1}^{(n)}, \rho_{g_2}^{(n)}, \cdots, \rho_{g_{n-s}}^{(n)})$, where $\{g_i\}_{i\in[n-s]} = [n]-\mathcal{S}$ and $g_1 < g_2 < \cdots < g_{n-s}$. Now, notice that $||\mathbf{G}^{(n-s,n)}_{\Gamma}||_F^2 = \sum_{i=1}^{n-s}\sum_{j=1}^{n-s}|T_{i-1}(\gamma_j)|^2$, and $|T_i(\gamma_j)| \leq 1$ for any $i, j \in [n-s]$. Therefore, we have

$$||\mathbf{G}^{(n-s,n)}_{\Gamma}||_F^2 \leq (n-s)^2. \tag{36}$$

In the following, we obtain an upper bound on $||(\mathbf{G}^{(n-s,n)}_{\Gamma})^{-1}||_F$. Let $L_{\Gamma,k}$ be the k-th Lagrange polynomial associated with Γ, that is,

$$L_{\Gamma,k}(x) = \prod_{i\in[n-s]-\{k\}}\frac{x-\gamma_i}{\gamma_k-\gamma_i} \tag{37}$$

Since $L_{\Gamma,k}(x)$ has a degree of $n-s-1$, it can be written in terms of the Chebyshev basis $T_0(x), \cdots, T_{n-s-1}(x)$ as

$$L_{\Gamma,k}(x) = \sum_{i=0}^{n-s-1} a_{i,k}T_i(x), \tag{38}$$

for some real coefficients $a_{0,k}, \cdots, a_{n-s-1,k}$. Now, from (37), note the following property regarding $L_{\Gamma,k}(x)$:

$$L_{\Gamma,k}(x) = \begin{cases} 1, & \text{if } x = \gamma_k \\ 0, & \text{if } x \in \{\gamma_i\}_{i\in[n-s]-k}. \end{cases}$$

Using this property and observing (38), we conclude that, for any $j \in [n-s]$, $\sum_{i=0}^{n-s-1} a_{i,k}T_i(\gamma_j) = \delta(k-j)$. Therefore,

$$\begin{pmatrix} a_{0,1} & \cdots & a_{n-s-1,1} \\ \vdots & \ddots & \vdots \\ a_{0,n-s} & \cdots & a_{n-s-1,n-s} \end{pmatrix}\mathbf{G}^{(n-s,n)}_{\Gamma} = \mathbf{I}_{n-s},$$

where $\mathbf{I}_{n-s}$ is the $n-s \times n-s$ identity matrix. That is,

$$\left(\mathbf{G}^{(n-s,n)}_{\Gamma}\right)^{-1} = \begin{pmatrix} a_{0,1} & \cdots & a_{n-s-1,1} \\ \vdots & \ddots & \vdots \\ a_{0,n-s} & \cdots & a_{n-s-1,n-s} \end{pmatrix}, \tag{39}$$

Therefore,

$$\left|\left|\left(\mathbf{G}^{(n-s,n)}_{\Gamma}\right)^{-1}\right|\right|_F^2 = \sum_{i=1}^{n-s}\sum_{j=1}^{n-s}|a_{i-1,j}|^2. \tag{40}$$

In addition, we have that

$$\begin{aligned}\sum_{k=1}^{n-s}\int_{-1}^{1}L_{\Gamma,k}^2(x)w(x)dx &= \sum_{k=1}^{n-s}\int_{-1}^{1}\sum_{i=0}^{n-s-1}\sum_{j=0}^{n-s-1}a_{i,k}a_{j,k}T_i(x)T_j(x)w(x)dx \\ &= \sum_{k=1}^{n-s}\sum_{i=0}^{n-s-1}\sum_{j=0}^{n-s-1}a_{i,k}a_{j,k}\int_{-1}^{1}T_i(x)T_j(x)w(x)dx \\ &= \sum_{k=1}^{n-s}\sum_{i=0}^{n-s-1}\sum_{j=0}^{n-s-1}a_{i,k}a_{j,k}\langle T_i, T_j\rangle \\ &= \sum_{k=1}^{n-s}\sum_{i=0}^{n-s-1}|a_{i,k}|^2. \end{aligned} \tag{41}$$

From (40) and (41), we conclude that $||(\mathbf{G}^{(n-s,n)}_{\Gamma})^{-1}||_F^2 = \sum_{k=1}^{n-s}\int_{-1}^{1}L_{\Gamma,k}^2(x)w(x)dx$. Now, we express the integral $\int_{-1}^{1}L_{\Gamma,k}^2(x)w(x)dx$ in the Gauss quadrature form using the n roots of $T_n(x)$: $\rho_1^{(n)}, \cdots, \rho_n^{(n)}$. Note that this is a "trick" we use in the proof - it is possible to use the Gauss quadrature formula over $n-s$ nodes to express the integral of the degree $2(n-s-1)$ polynomial $L_{\Gamma,k}^2(x)$. However, the use of n nodes instead of $n-s$ nodes leads to simple tractable bound for $||(\mathbf{G}^{(n-s,n)}_{\Gamma})^{-1}||_F^2$. Now, we can write

$$\int_{-1}^{1}L_{\Gamma,k}^2(x)w(x)dx = \sum_{i=1}^{n}c_iL_{\Gamma,k}^2(\rho_i^{(n)}), \tag{42}$$

for some constants $c_1, \cdots, c_n$. Moreover, $c_1, \cdots, c_n$ for the Chebyshev polynomials of the first kind are, in fact, all equal to π/n with respect to the weight function $w(x) = \frac{1}{\sqrt{1-x^2}}$ [29, Chapter 9]. Therefore, we have

$$\int_{-1}^{1}L_{\Gamma,k}^2(x)w(x)dx = \frac{\pi}{n}\sum_{i=1}^{n}L_{\Gamma,k}^2(\rho_i^{(n)}), \tag{43}$$

and, consequently,

$$\left|\left|\left(\mathbf{G}^{(n-s,n)}_{\Gamma}\right)^{-1}\right|\right|_F^2 = \frac{\pi}{n}\sum_{k=1}^{n-s}\sum_{i=1}^{n}L_{\Gamma,k}^2(\rho_i^{(n)}). \tag{44}$$

Now, from (37), note that $L_{\Gamma,k}(x)$ has the following evaluations

$$L_{\Gamma,k}(\rho_i^{(n)}) = \begin{cases} 1, & \text{if } i = g_k \\ 0, & \text{if } i \in \{g_i\}_{i\in[n-s]-k} \\ \prod_{j\in[n-s]-\{k\}}\frac{\rho_i^{(n)}-\gamma_j}{\gamma_k-\gamma_j}, & \text{if } i \in \mathcal{S}. \end{cases} \tag{45}$$

Therefore, (44) can be written as

$$\begin{aligned}&\left|\left|\left(\mathbf{G}_\Gamma^{(n-s,n)}\right)^{-1}\right|\right|_F^2\\ &=\frac{\pi}{n}\sum_{k=1}^{n-s}\left(1+\sum_{i\in\mathcal{S}}\prod_{j\in[n-s]-\{k\}}\left(\frac{\rho_i^{(n)}-\gamma_j}{\gamma_k-\gamma_j}\right)^2\right)\\ &=\frac{\pi(n-s)}{n}+\frac{\pi}{n}\sum_{k=1}^{n-s}\sum_{i\in\mathcal{S}}\prod_{j\in[n-s]-\{k\}}\left(\frac{\rho_i^{(n)}-\gamma_j}{\gamma_k-\gamma_j}\right)^2 \quad (46)\end{aligned}$$

In order to obtain our upper bound on $||(\mathbf{G}_\Gamma^{(n-s,n)})^{-1}||_F^2$, in the following, we get an upper bound on the term $\prod_{j\in[n-s]-\{k\}}\left(\frac{\rho_i^{(n)}-\gamma_j}{\gamma_k-\gamma_j}\right)^2$ in (46). Notice that $\prod_{j\in[n-s]-\{k\}}\left(\frac{\rho_i^{(n)}-\gamma_j}{\gamma_k-\gamma_j}\right)^2$ can be written as

$$\begin{aligned}&\prod_{j\in[n-s]-\{k\}}\left(\frac{\rho_i^{(n)}-\gamma_j}{\gamma_k-\gamma_j}\right)^2\\ &=\prod_{j\in[n-s]-\{k\}}\left(\frac{\rho_i^{(n)}-\rho_{g_j}^{(n)}}{\rho_{g_k}^{(n)}-\rho_{g_j}^{(n)}}\right)^2\\ &=\prod_{j\in[n-s]-\{k\}}\left(\frac{\rho_i^{(n)}-\rho_{g_j}^{(n)}}{\rho_{g_k}^{(n)}-\rho_{g_j}^{(n)}}\right)^2\\ &\quad\times\frac{\prod_{j\in\mathcal{S}\cup\{g_k\}-\{i\}}\left(\rho_i^{(n)}-\rho_j^{(n)}\right)^2}{\prod_{j\in\mathcal{S}\cup\{g_k\}-\{i\}}\left(\rho_i^{(n)}-\rho_j^{(n)}\right)^2}\\ &=\frac{1}{\prod_{j\in[n-s]-\{k\}}\left(\rho_{g_k}^{(n)}-\rho_{g_j}^{(n)}\right)^2}\\ &\quad\times\frac{\prod_{j\in[n]-\{i\}}\left(\rho_i^{(n)}-\rho_j^{(n)}\right)^2}{\prod_{j\in\mathcal{S}\cup\{g_k\}-\{i\}}\left(\rho_i^{(n)}-\rho_j^{(n)}\right)^2}\\ &=\frac{1}{\prod_{j\in[n-s]-\{k\}}\left(\rho_{g_k}^{(n)}-\rho_{g_j}^{(n)}\right)^2}\\ &\quad\times\frac{\left(2^{1-n}n/\sin(\frac{(2i-1)\pi}{2n})\right)^2}{\prod_{j\in\mathcal{S}\cup\{g_k\}-\{i\}}\left(\rho_i^{(n)}-\rho_j^{(n)}\right)^2}, \quad (47)\end{aligned}$$

where the last equality follows from Lemma B.1. Moreover, the product $\prod_{j\in[n-s]-\{k\}}\left(\rho_{g_k}^{(n)}-\rho_{g_j}^{(n)}\right)^2$ in (47) can be written as

$$\begin{aligned}\prod_{j\in[n-s]-\{k\}}\left(\rho_{g_k}^{(n)}-\rho_{g_j}^{(n)}\right)^2&=\prod_{j\in[n-s]-\{k\}}\left(\rho_{g_k}^{(n)}-\rho_{g_j}^{(n)}\right)^2\\ &\quad\times\frac{\prod_{j\in\mathcal{S}}\left(\rho_{g_k}^{(n)}-\rho_j^{(n)}\right)^2}{\prod_{j\in\mathcal{S}}\left(\rho_{g_k}^{(n)}-\rho_j^{(n)}\right)^2}\\ &=\frac{\left(2^{1-n}n/\sin(\frac{(2g_k-1)\pi}{2n})\right)^2}{\prod_{j\in\mathcal{S}}\left(\rho_{g_k}^{(n)}-\rho_j^{(n)}\right)^2}, \quad (48)\end{aligned}$$

where the last equality follows from Lemma B.1. Now, substituting from (48) in (47) yields

$$\begin{aligned}&\prod_{j\in[n-s]-\{k\}}\left(\frac{\rho_i^{(n)}-\gamma_j}{\gamma_k-\gamma_j}\right)^2\\ &=\frac{\left(\sin(\frac{(2g_k-1)\pi}{2n})\right)^2}{\left(\sin(\frac{(2i-1)\pi}{2n})\right)^2}\frac{\prod_{j\in\mathcal{S}}\left(\rho_{g_k}^{(n)}-\rho_j^{(n)}\right)^2}{\prod_{j\in\mathcal{S}\cup\{g_k\}-\{i\}}\left(\rho_i^{(n)}-\rho_j^{(n)}\right)^2}\\ &=\frac{\left(\sin(\frac{(2g_k-1)\pi}{2n})\right)^2}{\left(\sin(\frac{(2i-1)\pi}{2n})\right)^2}\frac{\prod_{j\in\mathcal{S}-\{i\}}\left(\rho_{g_k}^{(n)}-\rho_j^{(n)}\right)^2}{\prod_{j\in\mathcal{S}-\{i\}}\left(\rho_i^{(n)}-\rho_j^{(n)}\right)^2},\\ &\le\frac{\left(\sin(\frac{(2g_k-1)\pi}{2n})\right)^2}{\left(\sin(\frac{(2i-1)\pi}{2n})\right)^2}\left[\frac{\max_{j\in\mathcal{S}-\{i\}}\left(\rho_{g_k}^{(n)}-\rho_j^{(n)}\right)^2}{\min_{j\in\mathcal{S}-\{i\}}\left(\rho_i^{(n)}-\rho_j^{(n)}\right)^2}\right]^{s-1}\\ &=\frac{1}{\left(\sin(\frac{(2i-1)\pi}{2n})\right)^2}\left[\frac{4}{\left(\cos(\frac{\pi}{2n})-\cos(\frac{3\pi}{2n})\right)^2}\right]^{s-1}\\ &=\frac{4^{s-1}}{\left(\sin(\frac{(2i-1)\pi}{2n})\right)^2\left(\cos(\frac{\pi}{2n})-\cos(\frac{3\pi}{2n})\right)^{2(s-1)}}\\ &=O(4^{s-1}n^{2+4(s-1)}). \quad (49)\end{aligned}$$

Using (49) in (46), we conclude that

$$\left|\left|\left(\mathbf{G}_\Gamma^{(n-s,n)}\right)^{-1}\right|\right|_F^2=O(4^{s-1}(n-s)sn^{1+4(s-1)}). \quad (50)$$

Finally, combining (36) and (50), we conclude that

$$\kappa_F^{max}(\mathbf{G}^{(n-s,n)}(\rho^{(n)}))=O\left((n-s)\sqrt{ns(n-s)}\left(2n^2\right)^{s-1}\right).$$

□

Appendix C
Proof of Claim 7.2

Let $\alpha=2m_2-1,\gamma=\alpha(2m_1-1)$. $p_{\mathbf{A}}(x)$ in (31) can be written as

$$\begin{aligned}p_{\mathbf{A}}(x)&=\sum_{i=0}^{m_1-1}\sum_{j=0}^{m_2-1}\mathbf{A}_{i,j}T'_{m_2-1-j+i\alpha}(x)\\ &=\sum_{j=0}^{m_2-2}\mathbf{A}_{0,j}T_{m_2-1-j}(x)+1/2\ \mathbf{A}_{0,m_2-1}T_0(x)\\ &\quad+\sum_{i=1}^{m_1-1}\sum_{j=0}^{m_2-1}\mathbf{A}_{i,j}T_{m_2-1-j+i\alpha}(x). \quad (51)\end{aligned}$$

Similarly, $p_{\mathbf{B}}(x)$ in (31) can be written as

$$\begin{aligned}p_{\mathbf{B}}(x)&=\sum_{k=0}^{m_2-1}\sum_{l=0}^{m_3-1}\mathbf{B}_{k,l}T'_{k+l\gamma}(x)\\ &=1/2\ \mathbf{B}_{0,0}T_0(x)\\ &\quad+\sum_{k=1}^{m_2-1}\mathbf{B}_{k,0}T_k(x)+\sum_{k=0}^{m_2-1}\sum_{l=1}^{m_3-1}\mathbf{B}_{k,l}T_{k+l\gamma}(x) \quad (52)\end{aligned}$$

Now, the product $p_{\mathbf{A}}(x)p_{\mathbf{B}}(x)$ can be written as

$$p_{\mathbf{A}}(x)p_{\mathbf{B}}(x) = \frac{1}{2}\big(p_1(x) + p_2(x)\big) \tag{53}$$

where,

$$\begin{aligned}
p_1(x) = &\sum_{j=0}^{m_2-2} \mathbf{A}_{0,j}\mathbf{B}_{0,0}T_{m_2-1-j}(x) + \frac{1}{2}\mathbf{A}_{0,m_2-1}\mathbf{B}_{0,0}T_0(x)\\
&+ \sum_{i=1}^{m_1-1}\sum_{j=0}^{m_2-1} \mathbf{A}_{i,j}\mathbf{B}_{0,0}T_{m_2-1-j+i\alpha}(x)\\
&+ \sum_{j=0}^{m_2-2}\sum_{k=1}^{m_2-1} \mathbf{A}_{0,j}\mathbf{B}_{k,0}T_{m_2-1-j+k}(x)\\
&+ \sum_{k=1}^{m_2-1} \mathbf{A}_{0,m_2-1}\mathbf{B}_{k,0}T_k(x)\\
&+ \sum_{i=1}^{m_1-1}\sum_{j=0}^{m_2-1}\sum_{k=1}^{m_2-1} \mathbf{A}_{i,j}\mathbf{B}_{k,0}T_{m_2-1-j+i\alpha+k}(x)\\
&+ \sum_{j=0}^{m_2-2}\sum_{k=0}^{m_2-1}\sum_{l=1}^{m_3-1} \mathbf{A}_{0,j}\mathbf{B}_{k,l}T_{m_2-1-j+k+l\gamma}(x)\\
&+ \sum_{k=0}^{m_2-1}\sum_{l=1}^{m_3-1} \mathbf{A}_{0,m_2-1}\mathbf{B}_{k,l}T_{k+l\gamma}(x)\\
&+ \sum_{i=1}^{m_1-1}\sum_{j=0}^{m_2-1}\sum_{k=0}^{m_2-1}\sum_{l=1}^{m_3-1} \mathbf{A}_{i,j}\mathbf{B}_{k,l}T_{m_2-1-j+i\alpha+k+l\gamma}(x)
\end{aligned} \tag{54}$$

and,

$$\begin{aligned}
p_2(x) = &\sum_{j=0}^{m_2-2}\sum_{k=1}^{m_2-1} \mathbf{A}_{0,j}\mathbf{B}_{k,0}T_{|m_2-1-j-k|}(x)\\
&+ \sum_{i=1}^{m_1-1}\sum_{j=0}^{m_2-1}\sum_{k=1}^{m_2-1} \mathbf{A}_{i,j}\mathbf{B}_{k,0}T_{|m_2-1-j+i\alpha-k|}(x)\\
&+ \sum_{j=0}^{m_2-2}\sum_{k=0}^{m_2-1}\sum_{l=1}^{m_3-1} \mathbf{A}_{0,j}\mathbf{B}_{k,l}T_{|m_2-1-j-k-l\gamma|}(x)+\\
&\sum_{i=1}^{m_1-1}\sum_{j=0}^{m_2-1}\sum_{k=0}^{m_2-1}\sum_{l=1}^{m_3-1} \mathbf{A}_{i,j}\mathbf{B}_{k,l}T_{|m_2-1-j+i\alpha-k-l\gamma|}(x).
\end{aligned} \tag{55}$$

Now, in order to prove the claim, it suffices to prove the following two statements:

1) For any $i \in \{0,\cdots,m_1-1\}, l \in \{0,\cdots,m_3-1\}$, $\mathbf{C}_{i,l}$ is the matrix coefficient of $T_{m_2-1+i\alpha+l\gamma}$ in $p_1(x)$.
2) For any $i \in \{0,\cdots,m_1-1\}, l \in \{0,\cdots,m_3-1\}$, the matrix coefficient of $T_{m_2-1+i\alpha+l\gamma}$ in $p_2(x)$ is $\mathbf{0}_{N_1/m_1\times N_3/m_3}$, where $\mathbf{0}_{N_1/m_1\times N_3/m_3}$ is the $N_1/m_1 \times N_3/m_3$ all zeros matrix.

In the following, we prove that statement 1) is true. In order to find the coefficient of $T_{m_2-1+i\alpha+l\gamma}$ in $p_1(x)$, we find the set $\mathcal{S}_1 = \{(i',j',k',l') : m_2-1-j'+i'\alpha+k'+l'\gamma = m_2-1+i\alpha+l\gamma\}$. Rewriting $m_2-1-j'+i'\alpha+k'+l'\gamma = m_2-1+i\alpha+l\gamma$, we have

$$(k'-j') + (i'-i)\alpha + (l'-l)\gamma = 0. \tag{56}$$

(56) implies that $l' = l$. Suppose $l' \neq l$, this means that $(k'-j') + (i'-i)\alpha = c\gamma$ for some integer c. However, this is a contradiction since $|(k'-j')+(i'-i)\alpha| < \gamma$, for any i,i',j',k'. Now, (56) can be written as

$$(k'-j') + (i'-i)\alpha = 0. \tag{57}$$

Again, (57) implies $i' = i$. Suppose $i' \neq i$, this means $k'-j' = c\alpha$, for some integer c. However, this is a contradiction since $|k'-j'| < \alpha$. Now, since $i' = i$, (57) implies $j' = k'$. Thus, $\mathcal{S}_1 = \{(i,j',j',k) : j' \in \{0,\cdots,m_2-1\}\}$. That is, for any $i \in \{0,\cdots,m_1-1\}, j \in \{0,\cdots,m_3-1\}$, the matrix coefficient of $T_{m_2-1+i\alpha+l\gamma}$ in $p_1(x)$ is $\sum_{j'=0}^{m_2-1}\mathbf{A}_{i,j'}\mathbf{B}_{j',l} = \mathbf{C}_{i,l}$.

Now, it remains to prove statement 2). That is, for any $i \in \{0,\cdots,m_1-1\}, l \in \{0,\cdots,m_3-1\}$, the matrix coefficient of $T_{m_2-1+i\alpha+l\gamma}$ in $p_2(x)$ is $\mathbf{0}_{N_1/m_1\times N_3/m_3}$. In order to find the coefficient of $T_{m_2-1+i\alpha+l\gamma}$ in $p_2(x)$, we find the sets $\mathcal{S}_2^{(1)} = \{(i',j',k',l') : m_2-1-j'+i'\alpha-k'-l'\gamma = m_2-1+i\alpha+l\gamma\}$, and $\mathcal{S}_2^{(2)} = \{(i',j',k',l') : -m_2+1+j'-i'\alpha+k'+l'\gamma = m_2-1+i\alpha+l\gamma\}$.

First, for the set $\mathcal{S}_2^{(1)}$, rewriting $m_2-1-j'+i'\alpha-k'-l'\gamma = m_2-1+i\alpha+l\gamma$, we get

$$(-j'-k') + (i'-i)\alpha + (l+l')\gamma = 0. \tag{58}$$

From (58), we conclude that $l + l' = 0$. Otherwise, $(-j'-k') + (i'-i)\alpha = c\gamma$, for some integer c, a contradiction since $|(-j'-k') + (i'-i)\alpha| < \gamma$. Since $l + l' = 0$ and both l, l' are nonnegative, we conclude that $l' = l = 0$. Moreover, now (58) reduces to

$$(-j'-k') + (i'-i)\alpha = 0. \tag{59}$$

Again, since $|-j'-k'| < \alpha$, we conclude that $i' = i$, which implies that $j' + k' = 0$. Since $j' + k' = 0$ and both j', k' are nonnegative, we conclude that $j' = k' = 0$. Thus, $\mathcal{S}_2^{(1)} = \{(i,0,0,0)\}$. Now, noticing from (55) that $\mathbf{A}_{i,0}\mathbf{B}_{0,0}$ does not contribute to any term in $p_2(x)$, we conclude that the matrix coefficient of $T_{m_2-1+i\alpha+l\gamma}$ in $p_2(x)$ is only due to the set $\mathcal{S}_2^{(2)}$. Recall that $\mathcal{S}_2^{(2)} = \{(i',j',k',l') : -m_2+1+j'-i'\alpha+k'+l'\gamma = m_2-1+i\alpha+l\gamma\}$, we rewrite $-m_2+1+j'-i'\alpha+k'+l'\gamma = m_2-1+i\alpha+l\gamma$ as

$$(j'+k'-2m_2+2) - (i'+i)\alpha + (l'-l)\gamma = 0 \tag{60}$$

From (60), we conclude that $l = l'$. Otherwise, $(j'+k'-2m_2+2) - (i'+i)\alpha = c\gamma$, for some integer c, a contradiction since $|(j'+k'-2m_2+2) - (i'+i)\alpha| < \gamma$. Moreover, now (60) reduces to

$$(j'+k'-2m_2+2) + (i'+i)\alpha = 0. \tag{61}$$

Again, since $|j'+k'-2m_2+2| < \alpha$, we conclude that $i'+i = 0$. Since $i'+i = 0$ and both i, i' are nonnegative, we conclude that $i' = i = 0$, which implies that $j'+k' = 2m_2-2$. Since $j'+k' = 2m_2-2$ and both $j', k' \leq m_2-1$, we conclude that $j' = k' = m_2-1$. Thus, $\mathcal{S}_2^{(2)} = \{(0,m_2-1,m_2-1,l)\}$. Now, noting from (55) that $\mathbf{A}_{0,m_2-1}\mathbf{B}_{m_2-1,l}$ does not contribute to any term in $p_2(x)$, thus the matrix coefficient of $T_{m_2-1+i\alpha+l\gamma}$ in $p_2(x)$ is $\mathbf{0}_{N_1/m_1\times N_3/m_3}$. □

Appendix D
The Computational System's Forward Error Analysis

In the following, we conduct a forward error analysis for the Generalized OrthoMatDot codes when $m_1 = 1, m_2 = m, m_3 = 1$, for some positive integer m, and obtain an upper bound to the overall system's error at the output. In doing this, we follow the notation described in Section VII. In addition, the u-th row of $\mathbf{A}_{0,i}$, for any $i \in \{0,\cdots,m-1\}$, is denoted by $\mathbf{A}_{0,i}^{(u)}$, and the v-th column of $\mathbf{B}_{i,0}$, for any $i \in \{0,\cdots,m-1\}$, is denoted by $\mathbf{B}_{i,0}^{(v)}$. Moreover, let the vertical concatenation of $\mathbf{A}_{0,m-1}^{(u)},\cdots,\mathbf{A}_{0,0}^{(u)}$ be denoted $\mathbf{A}_0^{(u)}$ and the horizontal concatenation of $\mathbf{B}_{0,0}^{(v)},\cdots,\mathbf{B}_{m-1,0}^{(v)}$ be denoted $\mathbf{B}_0^{(v)}$, i.e.,

$$\mathbf{A}_0^{(u)} = \begin{pmatrix} \mathbf{A}_{0,m-1}^{(u)} \\ \mathbf{A}_{0,m-2}^{(u)} \\ \vdots \\ \mathbf{A}_{0,0}^{(u)} \end{pmatrix}, \quad \mathbf{B}_0^{(v)} = \begin{pmatrix} \mathbf{B}_{0,0}^{(v)} & \mathbf{B}_{1,0}^{(v)} & \dots & \mathbf{B}_{m-1,0}^{(v)} \end{pmatrix}, \tag{62}$$

for any $u \in [N_1]$ and $v \in [N_3]$. Similarly, for every worker $r \in [P]$, the u-th row of $p_{\mathbf{A}}(\rho_r^{(P)})$ is denoted by $p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})$, and the v-th column of $p_{\mathbf{B}}(\rho_r^{(P)})$ is denoted by $p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})$, i.e.,

$$p_{\mathbf{A}}(\rho_r^{(P)}) = \begin{pmatrix} p_{\mathbf{A}}^{(1)}(\rho_r^{(P)}) \\ p_{\mathbf{A}}^{(2)}(\rho_r^{(P)}) \\ \vdots \\ p_{\mathbf{A}}^{(N_1)}(\rho_r^{(P)}) \end{pmatrix},$$
$$p_{\mathbf{B}}(\rho_r^{(P)}) = \begin{pmatrix} p_{\mathbf{B}}^{(1)}(\rho_r^{(P)}) & p_{\mathbf{B}}^{(2)}(\rho_r^{(P)}) & \dots & p_{\mathbf{B}}^{(N_3)}(\rho_r^{(P)}) \end{pmatrix}. \tag{63}$$

Also, for any $\mathbf{x} = (x_1,\cdots,x_n) \in \mathbb{R}^n$, we define the $k \times n$ matrix $\mathbf{G}^{'(k,n)}(\mathbf{x})$ as:

$$\mathbf{G}^{'(k,n)}(\mathbf{x}) = \begin{pmatrix} T_0(x_1)/2 & \cdots & T_0(x_n)/2 \\ T_1(x_1) & \cdots & T_1(x_n) \\ \vdots & \ddots & \vdots \\ T_{k-1}(x_1) & \cdots & T_{k-1}(x_n) \end{pmatrix}, \tag{64}$$

and, letting $\mathcal{S} = (s_1,\cdots,s_r) \subseteq [n]$ such that $s_1 < \cdots < s_r$, we denote by $\mathbf{G}_{\mathcal{S}}^{'(k,n)}(\mathbf{x})$ the sub-matrix of $\mathbf{G}^{'(k,n)}(\mathbf{x})$ formed by concatenating columns with indices in $\mathcal{S}$, i.e.,

$$\mathbf{G}_{\mathcal{S}}^{'(k,n)}(\mathbf{x}) = \begin{pmatrix} T_0(x_{s_1})/2 & \cdots & T_0(x_{s_r})/2 \\ T_1(x_{s_1}) & \cdots & T_1(x_{s_r}) \\ \vdots & \ddots & \vdots \\ T_{k-1}(x_{s_1}) & \cdots & T_{k-1}(x_{s_r}) \end{pmatrix}. \tag{65}$$

A. Encoding Stage (Master Node)

Given the encoding polynomials in (31), at $m_1 = m_3 = 1, m_2 = m$, the master node sends

$$\begin{aligned} p_{\mathbf{A}}^{(u)}(\rho_r^{(P)}) &= \sum_{j=0}^{m-1} \mathbf{A}_{0,j}^{(u)} T'_{m-1-j}(\rho_r^{(P)}) \\ &= (\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)}))^{\mathcal{T}} \mathbf{A}_0^{(u)}, \quad \text{and} \\ p_{\mathbf{B}}^{(v)}(\rho_r^{(P)}) &= \sum_{k=0}^{m-1} \mathbf{B}_{k,0}^{(v)} T'_k(\rho_r^{(P)}) \\ &= \mathbf{B}_0^{(v)}\ \mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)}) \end{aligned} \tag{66}$$

to worker node r, for all $u, v \in [N_1], [N_3]$, respectively, for each $r \in [P]$. Consider a norm $||\cdot||$, we have

$$\begin{aligned} ||p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})|| &= ||(\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)}))^{\mathcal{T}} \mathbf{A}_0^{(u)}|| \\ &\le ||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||\ ||\mathbf{A}_0^{(u)}||. \end{aligned} \tag{67}$$

Let $\delta\mathbf{A}_0^{(u)}, \delta\mathbf{B}_0^{(v)}, \delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)}), \delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)}), \delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})$ be infinitesimal perturbations in $\mathbf{A}_0^{(u)}, \mathbf{B}_0^{(v)}$, $\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})$, $p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})$, $p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})$, respectively, we have

$$\begin{aligned} &p_{\mathbf{A}}^{(u)}(\rho_r^{(P)}) + \delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)}) \\ &= (\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)}) + \delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)}))(\mathbf{A}_0^{(u)} + \delta\mathbf{A}_0^{(u)}). \end{aligned} \tag{68}$$

Given (66), we conclude that

$$\begin{aligned} \delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)}) = \mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})\delta\mathbf{A}_0^{(u)} + \delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})\mathbf{A}_0^{(u)} \\ + \delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})\delta\mathbf{A}_0^{(u)}. \end{aligned} \tag{69}$$

Therefore,

$$\begin{aligned} ||\delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})|| &= ||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})\delta\mathbf{A}_0^{(u)} \\ &\quad + \delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})\mathbf{A}_0^{(u)} \\ &\quad + \delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})\delta\mathbf{A}_0^{(u)}|| \\ &\le ||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||\ ||\delta\mathbf{A}_0^{(u)}|| \\ &\quad + ||\delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||\ ||\mathbf{A}_0^{(u)}|| \\ &\quad + ||\delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||\ ||\delta\mathbf{A}_0^{(u)}||. \end{aligned} \tag{70}$$

Similarly,

$$\begin{aligned} ||p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})|| &= ||\mathbf{B}_0^{(v)}\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})|| \\ &\le ||\mathbf{B}_0^{(v)}||\ ||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||, \end{aligned} \tag{71}$$

and

$$\begin{aligned} ||\delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})|| &\le ||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||\ ||\delta\mathbf{B}_0^{(v)}|| \\ &\quad + ||\delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||\ ||\mathbf{B}_0^{(v)}|| \\ &\quad + ||\delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||\ ||\delta\mathbf{B}_0^{(v)}||. \end{aligned} \tag{72}$$

B. Computation Stage (Worker Nodes)

For every $r \in [P]$, worker node r receives $p_{\mathbf{A}}(\rho_r^{(P)})$ and $p_{\mathbf{B}}(\rho_r^{(P)})$ performs the matrix multiplication $p_{\mathbf{C}}(\rho_r^{(P)}) = p_{\mathbf{A}}(\rho_r^{(P)})p_{\mathbf{B}}(\rho_r^{(P)})$. Let $p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)})$ be the (u,v)-th entry of $p_{\mathbf{C}}(\rho_r^{(P)})$. Recalling (63), for any $u, v \in [N_1], [N_3]$, respectively, we can express $p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)})$ in the inner product form:

$$p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)}) = p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})\ p_{\mathbf{B}}^{(v)}(\rho_r^{(P)}). \tag{74}$$

Consider a norm $||\cdot||$, we have

$$\begin{aligned}||p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)})|| &= ||p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})\ p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})|| \\ &\leq ||p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})||\ ||p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})||. \end{aligned}\tag{75}$$

Let $\delta p_{\mathbf{A}}^{(u,v)}(\rho_r^{(P)}), \delta p_{\mathbf{B}}^{(u,v)}(\rho_r^{(P)}), \delta p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)})$ be infinitesimal perturbations in $p_{\mathbf{A}}^{(u,v)}(\rho_r^{(P)}), p_{\mathbf{B}}^{(u,v)}(\rho_r^{(P)}), p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)})$, respectively, we have

$$\begin{aligned}&p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)}) + \delta p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)}) \\ &= \left(p_{\mathbf{A}}^{(u)}(\rho_r^{(P)}) + \delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})\right)\left(p_{\mathbf{B}}^{(v)}(\rho_r^{(P)}) + \delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})\right).\end{aligned}\tag{76}$$

Given (74), we conclude that

$$\begin{aligned}\delta p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)}) &= p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})\delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)}) \\ &+ \delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})p_{\mathbf{B}}^{(v)}(\rho_r^{(P)}) \\ &+ \delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})\delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)}).\end{aligned}\tag{77}$$

Therefore,

$$\begin{aligned}||\delta p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)})|| &= ||p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})\delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)}) \\ &+ \delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})p_{\mathbf{B}}^{(v)}(\rho_r^{(P)}) \\ &+ \delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})\delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})|| \\ &\leq ||p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})||\ ||\delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})|| \\ &+ ||\delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})||\ ||p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})|| \\ &+ ||\delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})||\ ||\delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})||.\end{aligned}\tag{78}$$

C. Decoding Stage (Fusion Node)

Let $\mathcal{R} = \{r_1, r_2, \cdots, r_{2m-1}\}$ be the set of indices of the first $2m-1$ worker nodes to send their output to the fusion node. Thus, for all $u, v \in [N_1], [N_3]$, respectively, the fusion node receives $p_{\mathbf{C}}^{(u,v)}(\rho_{r_1}^{(P)}), \cdots, p_{\mathbf{C}}^{(u,v)}(\rho_{r_{2m-1}}^{(P)})$. Define the vector $p_{\mathbf{C}}^{(u,v)}(\rho_{\mathcal{R}}^{(P)}) = (p_{\mathbf{C}}^{(u,v)}(\rho_{r_1}^{(P)}) \ \cdots \ p_{\mathbf{C}}^{(u,v)}(\rho_{r_{2m-1}}^{(P)}))$. Next, the fusion node is required to solve the systems of linear equations: $\mathbf{c}^{(u,v)}\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}) = p_{\mathbf{C}}^{(u,v)}(\rho_{\mathcal{R}}^{(P)})$, for all $u \in [N_1], v \in [N_3]$, where the vector $\mathbf{c}^{(u,v)}$ is defined as $(c_0^{(u,v)} \ \cdots \ c_{2m-2}^{(u,v)})$. Consider a norm $||\cdot||$, by [20, Theorem 3.1], we conclude (73), shown at the bottom of the page.

D. The Overall System's Forward Error Bound

Here, we obtain an upper bound on the forward error of the overall computational system for the Generalized OrthoMatDot codes, when $m_1 = 1, m_2 = m, m_3 = 1, m \in \mathbb{N}^+$. The result is formulated in Theorem D.1 and implies that given a machine epsilon ϵ, and assuming inputs' relative errors of $O(\epsilon)$, and letting the norms of the inputs be $O(1/\epsilon)$, if $\kappa \ll 1/\epsilon$, where κ is the condition number of the appropriate decoding matrix, then relative error at the output is $O(\epsilon\kappa)$. Such observation confirms the fact that the conditioning at the decoding stage, i.e., minimizing the condition number of the possible decoding matrices is indeed the crucial factor in obtaining accurate outputs. We describe such implication in a more technical way after presenting our result in Theorem D.1.

Theorem D.1: Consider the matrix multiplication problem described in Section VII-A and computed using Construction 4, when $m_1 = 1, m_2 = m, m_3 = 1, m \in \mathbb{N}^+$, on the system defined in section VII-A. Define $\epsilon \in \mathbb{R}^+$, and assume $||\delta\mathbf{A}_0^{(u)}|| \leq \epsilon||\mathbf{A}_0^{(u)}||$, $||\delta\mathbf{B}_0^{(v)}|| \leq \epsilon||\mathbf{B}_0^{(v)}||$, $||\delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})|| \leq \epsilon||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||$, and $||\delta\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})|| \leq \epsilon||\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})||$, for any $u \in [N_1], v \in [N_3], r \in [P]$, and $\mathcal{R} \subset [P]$ such that $|\mathcal{R}| = 2m-1$, we have

$$\begin{aligned}||\delta\mathbf{c}^{(u,v)}||_2 \leq &\frac{\epsilon\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))}{1-\epsilon\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))} \\ &\times\left(\frac{(2+\epsilon)(2+2\epsilon+\epsilon^2)m}{\sqrt{2m-1}}||\mathbf{A}_0^{(u)}||_2\ ||\mathbf{B}_0^{(v)}||_2 \right.\\ &\left. + ||\mathbf{c}^{(u,v)}||_2\right).\end{aligned}\tag{79}$$

Proof: Since $||\delta\mathbf{A}_0^{(u)}||_2 \leq \epsilon||\mathbf{A}_0^{(u)}||_2$ and $||\delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2 \leq \epsilon||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2$, substituting in (70) yields

$$||\delta p_{\mathbf{A}}^{(u)}(\rho_r^{(P)})||_2 \leq \epsilon(2+\epsilon)||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2\ ||\mathbf{A}_0^{(u)}||_2. \tag{80}$$

Similarly, since $||\delta\mathbf{B}_0^{(v)}||_2 \leq \epsilon||\mathbf{B}_0^{(v)}||_2$ and $||\delta\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2 \leq \epsilon||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2$, substituting in (72) yields

$$||\delta p_{\mathbf{B}}^{(v)}(\rho_r^{(P)})||_2 \leq \epsilon(2+\epsilon)||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2\ ||\mathbf{B}_0^{(v)}||_2. \tag{81}$$

Now, recalling (67), (71), along with (80), (81), and substituting in (78), we conclude

$$\begin{aligned}||\delta p_{\mathbf{C}}^{(u,v)}(\rho_r^{(P)})||_2 &\leq \epsilon(2+\epsilon)(2+2\epsilon+\epsilon^2) \\ &\times ||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2^2\ ||\mathbf{A}_0^{(u)}||_2\ ||\mathbf{B}_0^{(v)}||_2.\end{aligned}\tag{82}$$

$$||\delta\mathbf{c}^{(u,v)}|| \leq \frac{\kappa(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))(||\delta p_{\mathbf{C}}^{(u,v)}(\rho_{\mathcal{R}}^{(P)})|| + ||\delta\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})||||\mathbf{c}^{(u,v)}||)}{||\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})|| - \kappa(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))||\delta\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})||}. \tag{73}$$

Since $||\delta \mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})||_F \leq \epsilon ||\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})||_F$, (73) can be bounded as

$$||\delta \mathbf{c}^{(u,v)}||_2 \leq \frac{\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))}{1-\epsilon\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))} \times \left(\frac{||\delta p_{\mathbf{C}}^{(u,v)}(\boldsymbol{\rho}_{\mathcal{R}}^{(P)})||_2}{||\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})||_F} + \epsilon ||\mathbf{c}^{(u,v)}||_2\right). \tag{83}$$

Also, substituting from (82) in (83) yields

$$||\delta \mathbf{c}^{(u,v)}||_2 \leq \frac{\epsilon\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))}{1-\epsilon\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))} \times \Bigg((2+\epsilon)(2+2\epsilon+\epsilon^2) \times \frac{||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2^2 \; ||\mathbf{A}_0^{(u)}||_2 \; ||\mathbf{B}_0^{(v)}||_2}{||\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})||_F} + ||\mathbf{c}^{(u,v)}||_2\Bigg). \tag{84}$$

Noting that $\sqrt{2m-1} \leq ||\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})||_F$ and $||\mathbf{G}_r^{'(m,P)}(\boldsymbol{\rho}^{(P)})||_2 \leq \sqrt{m}$, we have

$$||\delta \mathbf{c}^{(u,v)}||_2 \leq \frac{\epsilon\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))}{1-\epsilon\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))} \times \left(\frac{(2+\epsilon)(2+2\epsilon+\epsilon^2)m}{\sqrt{2m-1}}||\mathbf{A}_0^{(u)}||_2 \; ||\mathbf{B}_0^{(v)}||_2 + ||\mathbf{c}^{(u,v)}||_2\right). \tag{85}$$

□

E. Implication of Theorem D.1

As the parameter ϵ in Theorem D.1 represents the machine epsilon, a well conditioning of the decoding matrix $\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)}))$ such that $\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})) \ll 1/\epsilon$ yields an acceptable relative error at the output $||\delta \mathbf{c}^{(u,v)}||_2 / \max(\frac{m}{\sqrt{2m-1}}||\mathbf{A}_0^{(u)}||_2 \;\; ||\mathbf{B}_0^{(v)}||_2, ||\mathbf{c}^{(u,v)}||_2)$ that is $O(\epsilon\kappa_2(\mathbf{G}_{\mathcal{R}}^{(2m-1,P)}(\boldsymbol{\rho}^{(P)})))$ when $||\mathbf{A}_0^{(u)}||_2, ||\mathbf{B}_0^{(v)}||_2$ are $O(1/\epsilon)$.

Appendix E
Upper Bound on the Condition Number of Gaussian Matrices

We first introduce the following theorem from [35].

Theorem E.1: Let $\mathbf{A}$ be an $m \times m$ matrix, $m \geq 3$, and let the entries of $\mathbf{A}$ be independent and identically distributed standard Gaussian random variables. Then, for all $\alpha > 1$,

$$\Pr(\kappa_2(\mathbf{A}) > m\alpha) < \frac{5.6}{\alpha},$$

where $\kappa_2(\mathbf{A})$ is the condition number of $\mathbf{A}$ with respect to the matrix norm induced by ℓ_2.

As a consequence, in the following, we extend the result in Theorem E.1 to bound the condition number of every $m \times m$ sub-matrix of a random $m \times P$ matrix with $i.i.d$ standard Gaussian entries, $P \geq m$.

Proof of Theorem 9.1: For any subset $\mathcal{S} \subseteq \{1, 2, \ldots, P\}$, let $\mathbf{H}_{\mathcal{S}}$ denote the $|\mathcal{S}| \times m$ sub-matrix of $\mathbf{H}$ containing the columns $\mathbf{H}$ corresponding to $\mathcal{S}$, and let $s = P - m$. Then we have

$$\begin{aligned}
&\Pr\Big(\kappa_2^{max}(\mathbf{H}) > mP^{2s}\Big) \\
&\quad = \Pr\Big(\bigcup_{\mathcal{S}' \subset [P], |\mathcal{S}'|=m} (\kappa_2(\mathbf{H}_{\mathcal{S}'}) > mP^{2s})\Big) \\
&\quad \overset{(1)}{\leq} \sum_{\mathcal{S}' \subset [P], |\mathcal{S}'|=m} \Pr\left(\kappa_2(\mathbf{H}_{\mathcal{S}'}) > mP^{2s}\right) \\
&\quad \overset{(2)}{=} \binom{P}{s} \Pr\left(\kappa_2(\mathbf{H}_{\mathcal{S}'}) > mP^{2s}\right) \\
&\quad \overset{(3)}{<} P^s \frac{5.6}{P^{2s}} \\
&\quad = \frac{5.6}{P^s},
\end{aligned}$$

where (1) follows from the union bound, (2) is for any $\mathcal{S}' \subset [P]$ such that $|\mathcal{S}'| = m$, and (3) follows from the fact that $\binom{P}{s} \leq P^s$ and Theorem E.1. □

References

[1] 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 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 2100–2108.

[2] S. Dutta, V. Cadambe, and P. Grover, "'Short-dot': Computing large linear transforms distributedly using coded short dot products," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, pp. 6171–6193, Oct. 2019.

[3] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2017, pp. 1264–1270.

[4] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 278–301, Jan. 2020.

[5] 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 *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1585–1589. [Online]. Available: http://arxiv.org/abs/1811.10751

[6] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 4403–4413.

[7] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, D. Precup and Y. W. Teh, Eds. Sydney, NSW, Australia: International Convention Centre, Aug. 2017, pp. 3368–3376. [Online]. Available: http://proceedings.mlr.press/v70/tandon17a.html

[8] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, J. Dy and A. Krause, Eds. Stockholm, Sweden: Stockholmsmässan, Jul. 2018, pp. 5610–5619. [Online]. Available: http://proceedings.mlr.press/v80/ye18a.html

[9] Y. Yang, P. Grover, and S. Kar, "Coding for a single sparse inverse problem," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1575–1579. [Online]. Available: http://arxiv.org/abs/1706.00163.

[10] F. Haddadpour, Y. Yang, V. Cadambe, and P. Grover, "Cross-iteration coded computing," in *Proc. 56th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2018, pp. 196–203.

[11] R. K. Maity, A. S. Rawa, and A. Mazumdar, "Robust gradient descent via moment encoding and LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 2734–2738.
[12] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1215–1225.
[13] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Trans. Comput.*, vol. C-33, no. 6, pp. 518–528, Jun. 1984.
[14] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2418–2422.
[15] S. Li, S. M. M. Kalan, Q. Yu, M. Soltanolkotabi, and A. S. Avestimehr, "Polynomially coded regression: Optimal straggler mitigation via data encoding," 2018, *arXiv:1805.09934*. [Online]. Available: http://arxiv.org/abs/1805.09934
[16] W. Gautschi and G. Inglese, "Lower bounds for the condition number of Vandermonde matrices," *Numerische Math.*, vol. 52, no. 3, pp. 241–250, May 1987.
[17] W. Gautschi, "How (un) stable are Vandermonde systems?" in *Asymptotic and Computational Analysis*, R. Wong, Ed. New York, NY, USA: Dekker, 1990, pp. 193–210.
[18] W. Gautschi, "Norm estimates for inverses of Vandermonde matrices," *Numerische Math.*, vol. 23, no. 4, pp. 337–347, Aug. 1974.
[19] L. Reichel and G. Opfer, "Chebyshev-Vandermonde systems," *Math. Comput.*, vol. 57, no. 196, pp. 703–721, 1991.
[20] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*, vol. 37. Berlin, Germany: Springer, 2010.
[21] L. N. Trefethen, *Approximation Theory and Approximation Practice*. Philadelphia, PA, USA: SIAM, 2013, vol. 128.
[22] U. Sheth *et al.*, "An application of storage-optimal MatDot codes for coded matrix multiplication: Fast k-nearest neighbors estimation," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 1113–1120.
[23] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*. Philadelphia, PA, USA: SIAM, 1997.
[24] J. Demmel and P. Koev, "The accurate and efficient solution of a totally positive generalized Vandermonde linear system," *SIAM J. Matrix Anal. Appl.*, vol. 27, no. 1, pp. 142–152, Jan. 2005.
[25] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally decodable matrices for distributed matrix-vector multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1777–1781.
[26] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, Mar. 2020.
[27] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2022–2026.
[28] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.
[29] N. L. Carothers, "A short course on approximation theory," Dept. Math. Statist., Bowling Green State Univ., 1998.
[30] J. So, B. Guler, A. S. Avestimehr, and P. Mohassel, "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning," 2019, *arXiv:1902.00641*. [Online]. Available: http://arxiv.org/abs/1902.00641
[31] *MATLAB, Version 9.6.0 (R2019a)*, MathWorks, Natick, MA, USA 2019.
[32] M. Fahim and V. Cadambe. (2020). *Polynomially Coded Computing Repository*. [Online]. Available: https://github.com/mtarekf/poly-coded-computing
[33] A. Björck and V. Pereyra, "Solution of Vandermonde systems of equations," *Math. Comput.*, vol. 24, no. 112, pp. 893–903, 1970.
[34] H. V. D. Vel, "Numerical treatment of a generalized Vandermonde system of equations," *Linear Algebra Appl.*, vol. 17, no. 2, pp. 149–179, 1977. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0024379577900350
[35] J.-M. Azaïs and M. Wschebor, "Upper and lower bounds for the tails of the distribution of the condition number of a Gaussian matrix," *SIAM J. Matrix Anal. Appl.*, vol. 26, no. 2, pp. 426–440, Jan. 2004.
[36] A. Ganesan and P. O. Vontobel, "On the existence of universally decodable matrices," *IEEE Trans. Inf. Theory*, vol. 53, no. 7, pp. 2572–2575, Jul. 2007.

Mohammad Fahim (Student Member, IEEE) received the B.Sc. (Hons.) and the M.Sc. degrees in electrical engineering from Alexandria University, Alexandria, Egypt, in 2012 and 2015, respectively, and the Ph.D. degree in electrical engineering from Pennsylvania State University, University Park, PA, USA, in 2020.

He did an Internship with Texas A&M University, Qatar, on summer 2013. He was a Research Assistant with The American University in Cairo from 2012 to 2015 and a Teaching Assistant with Alexandria University, Egypt, from 2013 to 2015. He was a Graduate Research Assistant with the School of Electrical Engineering and Computer Science, Pennsylvania State University, from August 2015 to May 2020. He is currently a Senior Systems Engineer with the Wireless Research and Development, Qualcomm Technologies, San Diego, CA, USA. His research interests include space time codes in wireless communications, network coding, and secure, communication-efficient, and fault-tolerant codes for large scale distributed computing systems using tools from information theory and coding theory.

Viveck R. Cadambe (Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology Madras, Chennai, India, in 2006, and the Ph.D. degree in electrical and computer engineering from the University of California, Irvine, CA, USA, in 2011.

From 2011 and 2014, he was a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering (ECE), Boston University, and the Research Laboratory of Electronics (RLE), Massachusetts Institute of Technology (MIT). He is currently an Associate Professor with the Department of Electrical Engineering, Pennsylvania State University, University Park, PA, USA. His research uses tools of information and coding theory to understand fundamental questions in distributed computing and learning, cloud data storage, and communication networks.

Dr. Cadambe was a recipient of the 2009 IEEE Information Theory Society Best Paper Award, the 2011 CPCC Best Dissertation Award from the University of California, Irvine, the 2014 IEEE International Symposium on Network Computing and Applications (NCA) Best Paper Award, the 2015 NSF CRII Award, the 2016 NSF Career Award, the 2019 Google Faculty Research Award, and a finalist for the 2016 Bell Labs Prize. He served as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2014 to 2019. He serves as a Guest Editor for the IEEE JOURNAL OF SELECTED AREAS IN INFORMATION THEORY for the Special Issue on Coded Computing.