Securely Aggregated Coded Matrix Inversion

Neophytos Charalambides[μ], **Mert Pilanci**[σ]**, and Alfred O. Hero III**[μ]

[μ]EECS Department University of Michigan [σ]EE Department Stanford University

Email: `neochara@umich.edu`, `pilanci@stanford.edu`, `hero@umich.edu`

Abstract

Coded computing is a method for mitigating straggling workers in a centralized computing network, by using erasure-coding techniques. Federated learning is a decentralized model for training data distributed across client devices. In this work we propose approximating the inverse of an aggregated data matrix, where the data is generated by clients; similar to the federated learning paradigm, while also being resilient to stragglers. To do so, we propose a coded computing method based on gradient coding. We modify this method so that the coordinator does not access the local data at any point; while the clients access the aggregated matrix in order to complete their tasks. The network we consider is not centrally administrated, and the communications which take place are secure against potential eavesdroppers.

I. INTRODUCTION AND RELATED WORK

Inverting a matrix is one of the most important operations in numerous applications, such as, signal processing, machine learning, and scientific computing [1], [2]. A common way of inverting a matrix is to perform Gaussian elimination, which requires $\mathcal{O}(N^3)$ operations for square matrices of order N. In high-dimensional applications, this can be cumbersome. Over the past few years the machine learning (ML) community has made much progress on *federated learning* (FL), focusing on iterative methods.

The objective of FL is to leverage computation, communication and storage resources to perform distributed computations for ML models, where the data of each federated worker is never shared with the coordinator of the network; that aggregates local computations in order to update the model parameters. In FL applications it is important that the data is kept private and secure.

Distributed computations in the presence of *stragglers* (workers who fail to compute their task or have longer response times) must account for the effect of non-responsive workers. Coding-theoretic approaches have been adopted for this purpose [3], [4], and fall under the framework of *coded computing* (CC). Other techniques have also been utilized; to develop *approximate* CC schemes, *e.g.* equiangular tight frames [5] and sketching [6]. Data security is also an increasingly important issue in CC [7]. Despite the fact that multiplication algorithms imply inversion algorithms and vice versa, in the context of CC; matrix inversion has not been studied as extensively as *coded matrix multiplication* (CMM) [8]. The main reason for this is the fact that the latter is non-linear and non-parallelizable as an operator. We point out that distributed inversion algorithms do exist, though these make assumptions on the matrix, are specific for distributed and parallel computing platforms, and require a matrix factorization; or heavy and multiple communication instances between the workers and the coordinator.

In [9] a CC method[1] was proposed based on *gradient coding* (GC) [10], which approximates the inverse of a matrix $\mathbf{A}$. In order to overcome the obstacle of non-linearity, the columns of $\mathbf{A}^{-1}$ are approximated. When assuming infinite floating-point precision, this CCM introduces no numerical nor approximation errors. Note that GC and not CMM was utilized, as the latter does not require the encoding to be done locally by the workers.

Though the two areas of FL and CC seem to be closely related, on the surface they appear incompatible. For instance, in CC one often assumes there is a master server that distributes the data and may perform the encoding (encoding by the master server is done in CMM, but not in GC), while in FL the central coordinator never has access to the distributed local training data; which are located at different client nodes or workers.

There are a few recent works that leverage CC in order to devise secure FL methods for distributed regression and iterative optimization [11]–[16]. In this work, we combine optimization and CC, using erasure coding to protect against stragglers as in CC and locally approximating the inverse without revealing the data to the coordinator, to design a CCM which inverts a matrix from data aggregated through clients; and guarantees security against eavesdroppers. Our approach, is based on the *coded matrix inversion method* (CMIM) we develop, which utilizes *balanced Reed-Solomon* (BRS) codes [17], [18]. This results in an efficient decoding in terms of the threshold number of responsive workers needed to perform an error free computation. We show that the general class of *maximum distance separable* (MDS) generator matrices could be used to generate a suitable erasure code (Theorem 7). The focus is on BRS codes, which have the following advantages:

  (i) minimum redundancy per task across the network,
 (ii) they optimize communication from workers to the master,
(iii) we can efficiently decode the resulting method.

As noted in [11], [13], most CCMs are not applicable in FL. In our case, the obstacle is that all clients need to know each others data in order to invert the aggregated matrix, which we elaborate on in V-A. For this reason, we relax the privacy restriction of FL and allow the clients to recover the aggregated matrix $\mathbf{A}$, which is necessary and unavoidable for matrix inversion.

---

[1]We abbreviate 'coded computing method/methods' to CCM/CCMs.

Our CMIM can also be used to compute the Moore–Penrose pseudoinverse $\mathbf{Y}^\dagger$ of a data matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$ for $M \gg N$, which is more general than inverting a square matrix. By using the fact that $\mathbf{Y}^\dagger = (\mathbf{Y}^\top \mathbf{Y})^{-1} \mathbf{Y}^\top$, the bottleneck is computing the inverse of $\mathbf{A} = \mathbf{Y}^\top \mathbf{Y}$. In addition, two more matrix multiplications need to take place distributively: computing $\mathbf{A}$ before the inversion; and $\widehat{\mathbf{A}^{-1}} \mathbf{Y}^\top$ after the inverse has been approximated. The matrix products can be computed distributively using various CCMs, *e.g.* we can use a modification of the coded FL approaches of [13] and a CMM from [19]; both of which are based on GC. For the remainder of the paper, we focus on the generic problem of inverting a square matrix $\mathbf{A}$.

The proposed approach applies to general linear regression problems. Compared to traditional FL iterative approaches [20], the difference is that for $\mathbf{Y}\theta = \boldsymbol{p}$; with $\boldsymbol{p}$ the label vector and θ the model parameters, the pseudoinverse-regularized regression solution is $\hat{\theta} = \widehat{\mathbf{Y}^\dagger}\boldsymbol{p}$. Unlike conventional FL methods, this regularized regression can be computed non-iteratively. The non-iterative nature of the proposed approach is advantageous in settings such as Kalman filtering, where the matrix inverse must be updated in real time as measurements come in, as well as when dealing with time-series; and regularized regression with varying regularized coefficients.

We organize the paper as follows. In II we recall basic facts on matrix inversion, least squares approximation and finite fields. In III we review BRS codes, and prove two key lemmas regarding their generator matrices. In IV we present the matrix inverse approximation algorithm we utilize in our CCM. The main contribution is presented in V. Our approach is split into four phases, which we group in pairs of two. First, we discuss information sharing from the coordinator to the workers (we consider all the clients' servers as the network's workers), and then information sharing between the workers. Second, we show how our inversion algorithm can be incorporated in linear CCMs, and describe how this fits into the relaxed FL setting we are considering. Concluding remarks and future work are presented in VI.

A. Overview of the Coded Matrix Inversion Method

In CC the computational network is centralized, and is comprised of a master server who communicates with n workers. The idea behind our approximation is that the workers use a least squares solver to approximate multiple columns of $\mathbf{A}^{-1}$, resulting in a set of local approximations to submatrices of $\widehat{\mathbf{A}^{-1}}$, which we refer to as *blocks*. We present approximation guarantees and simulation results for *steepest descent* (SD) and *conjugate gradient* (CG) iterative optimization methods. By locally approximating the columns in this way, the workers can linearly encode the blocks of $\widehat{\mathbf{A}^{-1}}$. The clients have a block of data $\{\mathbf{A}_\iota\}_{\iota=1}^k$, which constitute the data matrix $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \cdots & \mathbf{A}_k \end{bmatrix}$. To simplify our presentation, we assume that each local data block is of the same size; *i.e.* $\mathbf{A}_\iota \in \mathbb{R}^{N \times T}$ for $T = N/k$, and that client i has n_i servers. Therefore, the total number of workers is $n = \sum_{j=1}^k n_j$. We assume the blocks are of the same size, so that the encodings carried out by the clients are consistent. In V, we show that this assumption is not necessary. Moreover, for the CCM, it is not required that the number of blocks equal the number of clients. For a given natural number γ, assume that γ divides T; denoted $\gamma \mid T$ (each local data block $\mathbf{A}_\iota$ is further divided into γ sub-blocks). In the case where $k \nmid N$ or $\gamma \nmid T$, we can pad the blocks of $\mathbf{A}^{-1}$ so that these assumptions are met.

A limitation of our proposed CMIM, is the fact that each worker needs to have full knowledge of $\mathbf{A}$, in order to estimate columns of $\mathbf{A}^{-1}$ through a least squares solver. The sensitivity of Gaussian elimination and matrix inversion also require that all clients have knowledge of each others' data [9]. This limitation is shared by other coded federated learning methods, *e.g.* CodedPaddedFL [13], and further justifies our requirement that task allocations need to be carefully distributed across the workers, especially in the context of FL. In contrast to CC and GC; where a master server has access to all the data, in FL the data is inherently distributed across devices, thus GC cannot be applied directly. We also assume that the coordinator does not intercept the communication between the clients, otherwise she could recover the local data. Also, we trust that the coordinator will not invert $\widehat{\mathbf{A}^{-1}}$, to approximate $\mathbf{A}$ — this would be computationally difficult, for N large.

Before broadcasting the data amongst themselves, the clients encode their block $\mathbf{A}_i$, which guarantees security from outside eavesdroppers. When the clients receive the encoded data, they can decrypt and recover $\mathbf{A}$. Then, their servers act as the workers of the proposed CMIM and carry out their assigned computations, and directly communicate their computations back to the coordinator. Once the *recovery threshold* (the minimum number of responses needed to recover the computation) is met, the approximation $\widehat{\mathbf{A}^{-1}}$ is recoverable.

B. Coded Federated Learning

There are few works that leverage CC to devise secure FL schemes. Most of these works have focused on distributed regression and iterative methods, which is the primary application for FL [11]–[15]. Below, we describe and compare these approaches to our work.

The authors of [11] proposed *coded federated learning*, in which they utilize a CMM scheme. Their security relies on the use of random linear codes, to define the parity data. Computations are carried out locally on the systematic data, and only the parity data is sent to the coordinator. The main drawback compared to our scheme is that each worker has to generate a random encoding matrix and apply a matrix multiplication for the encodings, while we use the same BRS generator matrix across the network, based on GC, to linearly encode the local computations. The drawback in our case, is that the workers need to securely share their data with each other. This is an artifact of the operation (inversion) we are approximating, and is inevitable in the general case

where $\mathbf{A}$ has no structure. Under the relaxed FL setting we are considering, where the data is gathered or generated locally and is not i.i.d., we cannot make any assumptions on the structure of $\mathbf{A}$.

In [13], two methods were proposed. CodedPaddedFL combines one-time-padding with GC to carry out the FL task. Some of its disadvantages are that a *one-time-pad* (OTP) needs to be generated by each worker, and that the OTPs are shared with the coordinator, which means that if she gets hold of the encrypted data, she can decrypt it, compromising security. Furthermore, there is a heavy communication load and the coordinator needs to store all the pads in order to recover the computed gradients. In the proposed CMIM, the coordinator generates a set of interpolation points, and shares them with the clients. If the coordinator can intercept the communication between the workers, she can decrypt the encrypted data blocks. The second method proposed in [13], CodedSecAgg, relies on *Shamir's secret sharing* (SSS); which is based on polynomial interpolation over finite fields. In contrast, our CMIM relies on GC and Lagrange interpolation.

Lastly, we discuss the method proposed in [15], which is based on the McEliece cryptosystem, and moderate-density parity-check codes. This scheme considers a communication delay model which defines stragglers as the workers who respond slower than the fastest worker, and time out after a predetermined amount of time Δ. As the iterative SD process carries on, such workers are continuously disregarded. Due to this, there is a data sharing step at each iteration, at which the new stragglers communicate encrypted versions of their data to the active workers. Our scheme is non-iterative, and has a fixed recovery threshold. Unlike some of the works previously mentioned, which guarantee information-theoretic security, the McEliece based systems and our approach have *computational* security guarantees.

C. Lagrange Interpolation and Polynomial CCMs

While there is extensive literature on matrix-vector and matrix-matrix multiplication, and computing the gradient in the presence of stragglers, there is limited work on computing or approximating the inverse of a matrix [21]. The non-linearity of matrix inversion prohibits linear or polynomial encoding of the data before the computations are to be performed. Consequently, most CCMs cannot be directly utilized. Gradient coding is the appropriate CC set up to consider [22], precisely because the encoding takes place once the computation has been completed, in contrast to most CMM methods where the encoding is done by the master, before the data is distributed. This helps improve the recovery threshold, which is a primary objective of the CMM problem.

Here, we give a brief overview of the GC scheme on which our CMIM is based. We also review "Lagrange Coded Computing" (LCC), which has relations to our approach. Then, we give a summary of our proposed CMIM. All these rely on Lagrange interpolation over finite fields. We then mention related CMM schemes based on Lagrange or polynomial interpolation.

Gradient codes are a class of codes designed to mitigate the effect of stragglers in data centers, by recovering the gradient of differentiable and additively separable objective functions in distributed first order methods [22]. The proposed CMIM utilizes BRS generator matrices constructed for GC [10]. The main difference from our work is that in GC the objective is to construct an encoding matrix $\mathbf{G} \in \mathbb{C}^{n \times k}$ and decoding vectors $\mathbf{a}_{\mathcal{I}} \in \mathbb{C}^k$, such that $\mathbf{a}_{\mathcal{I}}^{\top} \mathbf{G}_{\mathcal{I}} = \vec{\mathbf{1}}$ for any set of non-straggling workers indexed by $\mathcal{I}$. To do so, the decomposition of the BRS generator matrices $\mathbf{G}_{\mathcal{I}} = \mathbf{H}_{\mathcal{I}} \mathbf{P}$ is exploited, where $\mathbf{H}_{\mathcal{I}}$ is a Vandermonde matrix; and the first row of $\mathbf{P}$ is equal to $\vec{\mathbf{1}}$. Subsequently $\mathbf{a}_{\mathcal{I}}^{\top}$ is extracted as the first row of $\mathbf{H}_{\mathcal{I}}^{-1}$.

In the proposed CMIM framework, the objective is to design an *encoding-decoding pair* $(\tilde{\mathbf{G}}, \tilde{\mathbf{D}}_{\mathcal{I}})$ for which $\tilde{\mathbf{D}}_{\mathcal{I}} \tilde{\mathbf{G}}_{\mathcal{I}} = \mathbf{I}_N$, for all $\mathcal{I} \subsetneq \mathbb{N}_n$ of size k. The essential reason for requiring this condition, as opposed to that of GC, is that the empirical gradient of a given dataset is the sum of each individual gradients, while in our scenario if the columns of $\widehat{\mathbf{A}^{-1}}$ are summed; they cannot then be recovered.

The state-of-the art CC framework is LCC, which is used to compute arbitrary multivariate polynomials of a given dataset [4]; and has since been considered in various settings [23]–[27]. This approach is based on Lagrange interpolation, and it achieves the optimal trade-off between resiliency, security, and privacy. The problem we are considering is not a multivariate polynomial in terms of $\mathbf{A}$. To securely communicate $\mathbf{A}$ to the workers, we encode it through Lagrange interpolation. Though similar ideas appear in LCC, the purpose and application of the interpolation is different. Furthermore, LCC is a *point-based* approach and requires additional interpolation and linear combination steps after the decoding takes place, while ours is a *coefficient-based* CCM [28].

Recall that the workers in the CMIM must compute blocks of $\widehat{\mathbf{A}^{-1}}$. Once they complete their computations, they encode them by computing a linear combination with coefficients determined by a sparsest-balanced MDS generator matrix. Referring to the advantages claimed for CMIM in Section I, working with MDS generator matrices allows us to meet points (i) and (ii), while BRS generator matrices also helps us satisfy (iii). Once the recovery threshold is met, the coordinator can recover the approximation $\widehat{\mathbf{A}^{-1}}$. The structure of sparsest-balanced generator matrices is also leveraged to optimally allocate tasks to the workers, while linear encoding is what allows minimal communication load from the workers to the master. Security against eavesdroppers is guaranteed by encoding the local data through a modified Lagrange interpolation polynomial, before it is shared by the clients. This CMIM also extends to approximating $\mathbf{A}^{\dagger}$ [9].

Some of the earliest interpolation based CMM schemes are the Polynomial [8] and MatDot codes [29], both of which are point-based. The construction of 'Polynomial Codes' has since been generalized to 'Entangled Polynomial Codes' [30], which define similar polynomials to ours (12), though their use differs. We use (12) to encrypt the clients' data blocks; and our decryption is an evaluation at a finite field point. For Entangled Polynomial Codes two such polynomials are defined; one for each input matrix, and their product determines another degree $R - 2$ polynomial which is evaluated by each worker at a different point, before proceeding to the decoding step.

In MatDot codes [29] two polynomials are defined, one corresponding to each input, where instead of the Lagrange polynomial in (12); a monic monomial is multiplied by the partitions of the respective block submatrices. Then, analogous steps to those of Entangled Polynomial Codes take place, in order to recover the matrix product.

II. Preliminary Background

The set of $N \times N$ invertible matrices is denoted by $\mathrm{GL}_N(\mathbb{R})$. Recall that $\mathbf{A} \in \mathrm{GL}_N(\mathbb{R})$ has a unique inverse $\mathbf{A}^{-1}$, such that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_N$. The simplest way of computing $\mathbf{A}^{-1}$ is by performing Gaussian elimination on $\begin{bmatrix}\mathbf{A}|\mathbf{I}_N\end{bmatrix}$, which gives $\begin{bmatrix}\mathbf{I}_N|\mathbf{A}^{-1}\end{bmatrix}$ in $\mathcal{O}(N^3)$ operations. In Algorithm 1, we approximate $\mathbf{A}^{-1}$ column-by-column. We denote the ith row and column of $\mathbf{A}$ respectively by $\mathbf{A}_{(i)}$ and $\mathbf{A}^{(i)}$. The *condition number* of $\mathbf{A}$ is $\kappa_2 = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2$. The largest, smallest and i^{th} singular values of $\mathbf{A}$ are denoted by $\sigma_{\min}(\mathbf{A})$, $\sigma_{\max}(\mathbf{A})$ and $\sigma_i(\mathbf{A})$ respectively. For $\mathcal{I}$ an index subset of the rows of a matrix $\mathbf{M}$, the matrix consisting only of the rows indexed by $\mathcal{I}$, is denoted by $\mathbf{M}_{\mathcal{I}}$. We denote the set of integers between 1 and ν by $\mathbb{N}_\nu$. The support of a vector $\mathbf{v}$ is denoted by $\mathrm{supp}(\mathbf{v})$, and the number of nonzero elements of $\mathbf{A}$ by $\mathrm{nnzr}(\mathbf{A})$.

In the proposed algorithm we approximate N instances of the least squares minimization problem

$$\theta_{ls}^\star = \arg\min_{\theta \in \mathbb{R}^M} \left\{ \|\mathbf{A}\theta - \mathbf{y}\|_2^2 \right\} \tag{1}$$

for $\mathbf{A} \in \mathbb{R}^{N \times M}$ and $\mathbf{y} \in \mathbb{R}^N$. In many applications $N \gg M$, where the rows represent the feature vectors of a dataset. This has the closed-form solution $\theta_{ls}^\star = \mathbf{A}^\dagger \mathbf{y}$.

Computing $\mathbf{A}^\dagger$ to solve (1) is intractable for large M, as it requires computing the inverse of $\mathbf{A}^\top \mathbf{A}$. Instead, we use gradient methods to get *approximate* solutions, *e.g.* SD or CG, which require less operations, and can be done distributively. One could use second-order methods; *e.g.* Newton–Raphson, Gauss-Newton, Quasi-Newton, BFGS, or Krylov subspace methods instead. This would be worthwhile future work.

When considering a minimization problem with a convex differentiable objective function $\psi\colon \Theta \to \mathbb{R}$ over an open convex set $\Theta \subseteq \mathbb{R}^M$, as in (1), the SD procedure selects an initial $\theta^{[0]} \in \Theta$, and then updates θ according to:

$$\theta^{[t+1]} = \theta^{[t]} - \xi_t \cdot \nabla_\theta \psi(\theta^{[t]}), \quad \text{for } t = 0, 1, 2, \ldots$$

until a termination criterion is met, for ξ_t the step-size. The CG method is the most used and prominent iterative procedure for numerically solving systems of positive-definite equations.

Our proposed coding scheme is defined over the multiplicative cyclic group $(\mathbb{F}_q^\times, \cdot)$, for $\mathbb{F}_q$ the finite field of q elements and $\mathbb{F}_q^\times = \mathbb{F}_q \backslash \{0_{\mathbb{F}_q}\}$ its set of units. For implementation purposes, we identify $\mathbb{F}_q^\times$ with its realization in $\mathbb{C}$ as a subgroup of the circle group, since we assume our data is over $\mathbb{R}$. All operations can therefore be carried out over $\mathbb{C}$. Specifically, we identify β as an arbitrary primitive generator of $(\mathbb{F}_q^\times, \cdot)$. One such case is to identify $\beta \mapsto e^{2\pi i/(q-1)}$. Thus, for all $j \in \mathbb{N}_{q-1}$; we identify $\beta^j \mapsto e^{2\pi ij/(q-1)}$.

III. Balanced Reed-Solomon Codes

A Reed-Solomon code $\mathsf{RS}_q[n,k]$ over $\mathbb{F}_q$ for $q > n > k$, is the encoding of polynomials of degree at most $k-1$, for k the message length and n the code length. It represents our message over the *defining set of points* $\mathcal{A} = \{\alpha_j\}_{j=1}^n \subset \mathbb{F}_q$

$$\mathsf{RS}_q[n,k] = \Big\{ \big[f(\alpha_1), f(\alpha_2), \cdots, f(\alpha_n)\big] \ \Big| $$
$$f(X) \in \mathbb{F}_q[X] \text{ of degree } \leqslant k-1 \Big\}$$

where $\alpha_j = \alpha^j$, for α a primitive root of $\mathbb{F}_q$. Hence, each α_i is distinct. A natural interpretation of $\mathsf{RS}_q[n,k]$ is through its encoding map. Each message $(m_0, ..., m_{k-1}) \in \mathbb{F}_q^k$ is interpreted as $f(\mathsf{x}) = \sum_{i=0}^{k-1} m_i \mathsf{x}^i \in \mathbb{F}_q[\mathsf{x}]$, and f is evaluated at each point of $\mathcal{A}$. From this, $\mathsf{RS}_q[n,k]$ can be defined through the generator matrix

$$\mathbf{G} = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{k-1} \end{pmatrix} \in \mathbb{F}_q^{n \times k},$$

thus, RS codes are linear codes over $\mathbb{F}_q$. Furthermore, they attain the Singleton bound, *i.e.* $d = n - k + 1$, where d is the code's distance, which implies that they are MDS.

Balanced Reed-Solomon codes [17], [18] are a family of linear MDS error-correcting codes with generator matrices $\mathbf{G} \in \mathbb{F}_q^{n \times k}$ that are:

- **sparsest**: each *column* has the least possible number of nonzero entries
- **balanced**: each *row* contains the same number of nonzero entries

for the given code parameters k and n. The design of these generators are suitable for our purposes, as:

1) we have balanced loads across homogeneous workers,

2) sparse generator matrices reduce the computation tasks across the network,
3) the MDS property permits an efficient decoding step,
4) linear codes produce a compressed representation of the encoded blocks.

A. Balanced Reed-Solomon Codes for CC

In the proposed CMIM, we leverage BRS generator matrices to approximate $\mathbf{A}^{-1}$ distributively. For simplicity, we will consider the case where $d = s + 1 = \frac{nw}{k}$ is a positive integer[2], for n the number of workers and s the number of stragglers. Furthermore, d is the distance of the code and $\|\mathbf{G}^{(j)}\|_0 = d$ for all $j \in \mathbb{N}_k$; $\|\mathbf{G}_{(i)}\|_0 = w$ for all $i \in \mathbb{N}_n$, and $d > w$ since $n > k$. For decoding purposes, we require that at least $k = n - s$ workers respond. Consequently, $d = s + 1$ implies that $n - d = k - 1$. For simplicity, we also assume $d \geqslant n/2$. In our setting, each column of $\mathbf{G}$ corresponds to a computation task of computing a block of $\widehat{\mathbf{A}^{-1}}$; which we will denote by $\hat{\mathcal{A}}_i$, and each row corresponds to a worker.

Our choice of such a generator matrix $\mathbf{G} \in \mathbb{F}_q^{n \times k}$, solves

$$\arg \min_{\mathbf{G} \in \mathbb{F}_q^{n \times k}} \quad \left\{ \mathrm{nnzr}(\mathbf{G}) \right\}$$
$$\text{s.t.} \quad \|\mathbf{G}_{(i)}\|_0 \geqslant w, \ \forall i \in \mathbb{N}_n$$
$$\|\mathbf{G}^{(j)}\|_0 \geqslant d, \ \forall j \in \mathbb{N}_k \tag{2}$$
$$\mathrm{rank}(\mathbf{G}_\mathcal{I}) = k, \ \forall \mathcal{I} : |\mathcal{I}| = k$$

which determines an optimal task allocation among the workers of the proposed CMIM. The first and second constraints are analogous to the bound of [22, Theorem 1], which is met with equality in "perfectly balanced GC schemes". This theorem states that if all rows of $\mathbf{G}$ have the same number of nonzeros, then $\|\mathbf{G}_{(i)}\|_0 \geqslant k(s+1)/n$, for all i. By construction, the generator matrix $\mathbf{G}$ we propose, meets the first and second constraints with equality, for all $i \in \mathbb{N}_n$ and $j \in \mathbb{N}_k$.

Under the above assumptions, the entries of the generator matrix of a $\mathrm{BRS}_q[n, k]$ code meet the following:

- each column is sparsest, with exactly d nonzero entries
- each row is balanced, with $w = \frac{dk}{n}$ nonzero entries

where d equals to the number of workers who are tasked to compute each block, and w is the number of blocks that are computed by each worker.

Each column $\mathbf{G}^{(j)}$ corresponds to a polynomial $p_j(\mathsf{x})$, whose entries are the evaluation of the polynomial we define in (3) at each of the points of the defining set $\mathcal{A}$, i.e. $\mathbf{G}_{ij} = p_j(\alpha_i)$ for $(i, j) \in \mathbb{N}_n \times \mathbb{N}_k$. To construct the polynomials $\{p_j(\mathsf{x})\}_{j=1}^k$, for which $\deg(p_j) \leqslant \mathrm{nnzr}(\mathbf{G}^{(j)}) = n - d = k - 1$, we first need to determine a sparsest and balanced *mask matrix* $\mathbf{M} \in \{0, 1\}^{n \times k}$, which is ρ-sparse for $\rho = \frac{d}{n}$; i.e. $\mathrm{nnzr}(\mathbf{G}) = \rho nk$. We use the construction from [10], though it is fairly easy to construct more general such matrices, by using the Gale-Ryser Theorem [31], [32]. Even though this was not pointed out in [10], their construction of $\mathbf{M}$ (Algorithm 2) does not always produce a mask matrix of the given parameters when we select $d < n/2$. This is why in our work we require $d \geqslant n/2$. Furthermore, deterministic constructions resemble generator matrices of cyclic codes.

For our purposes we use $\mathcal{B} = \{\beta_j\}_{j=1}^n$ as our defining set of points, where each point corresponds to the worker with the same index. The objective now is to devise the polynomials $p_j(\mathsf{x})$, for which $p_j(\beta_i) = 0$ if and only if $\mathbf{M}_{ij} = 0$. Therefore:

(I) $\mathbf{M}_{ij} = 0 \implies (\mathsf{x} - \beta_i) \mid p_j(\mathsf{x})$
(II) $\mathbf{M}_{ij} \neq 0 \implies p_j(\beta_i) \in \mathbb{F}_q^\times$
for all pairs (i, j).

The construction of $\mathrm{BRS}_q[n, k]$ from [17] is based on what the authors called *scaled polynomials*. Below, we summarize the polynomial construction based on Lagrange interpolation [10]. We then prove a simple but important result that allows us to efficiently perform the decoding step.

The univariate polynomials corresponding to each column $\mathbf{G}^{(j)}$, are defined as:

$$p_j(\mathsf{x}) \coloneqq \prod_{i : \mathbf{M}_{ij} = 0} \left(\frac{\mathsf{x} - \beta_i}{\beta_j - \beta_i} \right) = \sum_{\iota=1}^k p_{j,\iota} \cdot \mathsf{x}^{\iota-1} \in \mathbb{F}_q[\mathsf{x}] \tag{3}$$

which satisfy (I) and (II). By the BCH bound [33, Chapter 9], it follows that $\deg(p_j) \geqslant n - d = k - 1$ for all $j \in \mathbb{N}_k$. Since each $p_j(\mathsf{x})$ is the product of $n - d$ monomials, we conclude that the bound on the degree is satisfied and met with equality, hence $p_{j,\iota} \in \mathbb{F}_q^\times$ for all coefficients.

By construction, $\mathbf{G}$ is decomposable into a Vandermonde matrix $\mathbf{H} \in \mathcal{B}^{n \times k}$ and a matrix comprised of the polynomial coefficients $\mathbf{P} \in (\mathbb{F}_q^\times)^{k \times k}$ [10]. Specifically, $\mathbf{G} = \mathbf{H}\mathbf{P}$ where $\mathbf{H}_{ij} = \beta_i^{j-1} = \beta^{i(j-1)}$ and $\mathbf{P}_{ij} = p_{j,i}$ are the coefficients from (3). This can be interpreted as $\mathbf{P}^{(j)}$ defining the polynomial $p_j(\mathsf{x})$, and $\mathbf{H}_{(i)}$ is comprised of the first k positive powers of β_i in ascending order, therefore

$$p_j(\beta_i) = \sum_{\iota=1}^k p_{j,\iota} \cdot \beta_i^{\iota-1} = \langle \mathbf{H}_{(i)}, \mathbf{P}^{(j)} \rangle.$$

---

[2]The case where $\frac{nw}{k} \in \mathbb{Q}_+ \backslash \mathbb{Z}_+$ is analyzed in [10], and also applies to our approach. We restrict our discussion to the case where $\frac{nw}{k} \in \mathbb{Z}_+$.

The following lemmas will help us respectively establish in our CC setting the efficiency of our decoding step and the optimality of the allocated tasks to the workers. For Lemma 1, recall that efficient matrix multiplication algorithms have complexity $\mathcal{O}(N^\omega)$, for $\omega < 2.372$ the *matrix multiplication exponent* [34].

Lemma 1. *The restriction $\mathbf{G}_\mathcal{I} \in \mathbb{F}_q^{k \times k}$ of $\mathbf{G}$ to any of its k rows indexed by $\mathcal{I} \subsetneq \mathbb{N}_n$, is an invertible matrix. Moreover, its inverse can be computed online in $\mathcal{O}(k^2 + k^\omega)$ operations.*

Proof. The matrices $\mathbf{H}$ and $\mathbf{P}$ are of size $n \times k$ and $k \times k$ respectively. The restricted matrix $\mathbf{G}_\mathcal{I}$ is then equal to $\mathbf{H}_\mathcal{I}\mathbf{P}$, where $\mathbf{H}_\mathcal{I} \in \mathbb{F}_q^{k \times k}$ is a square Vandermonde matrix, which is invertible in $\mathcal{O}(k^2)$ time [35]. Specifically

$$\mathbf{H}_\mathcal{I} = \begin{pmatrix} 1 & \beta_{\mathcal{I}_1} & \beta_{\mathcal{I}_1}^2 & \cdots & \beta_{\mathcal{I}_1}^{k-1} \\ 1 & \beta_{\mathcal{I}_2} & \beta_{\mathcal{I}_2}^2 & \cdots & \beta_{\mathcal{I}_2}^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \beta_{\mathcal{I}_k} & \beta_{\mathcal{I}_k}^2 & \cdots & \beta_{\mathcal{I}_k}^{k-1} \end{pmatrix} \in \mathbb{F}_q^{k \times k}.$$

It follows that

$$\det(\mathbf{H}_\mathcal{I}) = \prod_{\{i<j\} \subseteq \mathcal{I}} (\beta_j - \beta_i)$$

which is nonzero, since β is primitive. Therefore, $\mathbf{H}_\mathcal{I}$ is invertible. By [17, Lemma 1] and the BCH bound, we conclude that $\mathbf{P}$ is also invertible. Hence, $\mathbf{G}_\mathcal{I}$ is invertible for any set $\mathcal{I}$.

Note that the inverse of $\mathbf{P}$ can be computed a priori by the master before we deploy our CCM. Therefore, computing $\mathbf{G}_\mathcal{I}^{-1}$ online with knowledge of $\mathbf{P}^{-1}$, requires an inversion of $\mathbf{H}_\mathcal{I}$ which takes $\mathcal{O}(k^2)$ operations; and then multiplying it by $\mathbf{P}^{-1}$. Thus, it requires $\mathcal{O}(k^2 + k^\omega)$ operations in total. ∎

Lemma 2. *The generator matrix $\mathbf{G} \in \mathbb{F}_q^{n \times k}$ of a $\mathsf{BRS}_q[n, k]$ MDS code defined by the polynomials $p_j(\mathsf{x})$ of (3), solves the optimization problem* (2).

Proof. The first two constraints are satisfied by the construction of $\mathbf{G}$, which meets the sparsest and balanced constraints with equality; for the given parameters. The last constraint is implied by the MDS theorem, which states that every set of k rows of $\mathbf{G}$ is linearly independent.

The sparsity constraints of (2) imply that $\mathrm{nnzr}(\mathbf{G}) \geqslant \max\{nw, kd\}$, and for our parameters we have $nw = kd$. Both the first and second constraints are met with equality for the chosen $\mathbf{G}$. Moreover

$$\begin{aligned} \mathrm{nnzr}(\mathbf{G}) &= \sum_{j \in \mathbb{N}_k} \mathrm{nnzr}(\mathbf{G}^{(j)}) \\ &= \sum_{j \in \mathbb{N}_k} \#\{p_j(\beta_i) \neq 0 : \beta_i \in \mathcal{B}\} \\ &= \sum_{j \in \mathbb{N}_k} n - \{i : \mathbf{M}_{ij} = 0\} \\ &= \sum_{j \in \mathbb{N}_k} n - (n - d) \\ &= kd \end{aligned}$$

and the proof is complete. ∎

We conclude this subsection by recalling how the decomposition $\mathbf{G} = \mathbf{HP}$ is utilized for GC [10]. Each column of $\mathbf{G}$ corresponds to a partition of the data whose partial gradient is to be computed. The polynomials are judiciously constructed in this scheme, such that the constant term of each polynomial is 1, thus $\mathbf{P}_{(1)} = \vec{\mathbf{1}}$. By this, the decoding vector $\mathbf{a}_\mathcal{I}^\top$ is the first row of $\mathbf{H}_\mathcal{I}^{-1}$, for which $\mathbf{a}_\mathcal{I}^\top \mathbf{H}_\mathcal{I} = \mathbf{e}_1^\top$. A direct consequence of this is that $\mathbf{a}_\mathcal{I}^\top \mathbf{G}_\mathcal{I} = \mathbf{e}_1^\top \mathbf{P} = \mathbf{P}_{(1)} = \vec{\mathbf{1}}$, which is the objective for constructing a GC scheme.

IV. INVERSE APPROXIMATION ALGORITHM

Our goal is to estimate $\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_N \end{bmatrix}$, for $\mathbf{A}$ a square matrix of order N. A key property to note is

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}\begin{bmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_N \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{b}_1 & \cdots & \mathbf{A}\mathbf{b}_N \end{bmatrix} = \mathbf{I}_N$$

which implies that $\mathbf{A}\mathbf{b}_i = \mathbf{e}_i$ for all $i \in \mathbb{N}_N$, where $\mathbf{e}_i$ are the standard basis column vectors. Assume for now that we use any black-box least squares solver to estimate

$$\hat{\mathbf{b}}_i \approx \arg\min_{\mathbf{b} \in \mathbb{R}^N} \left\{ g_i(\mathbf{b}) := \|\mathbf{A}\mathbf{b} - \mathbf{e}_i\|_2^2 \right\} \tag{4}$$

which we call N times, to recover $\widehat{\mathbf{A}^{-1}} := \begin{bmatrix} \hat{\mathbf{b}}_1 & \cdots & \hat{\mathbf{b}}_N \end{bmatrix}$. This approach may be viewed as approximating

$$\widehat{\mathbf{A}^{-1}} \approx \arg \min_{\mathbf{B} \in \mathbb{R}^{N \times N}} \left\{ \|\mathbf{AB} - \mathbf{I}_N\|_F^2 \right\}.$$

Alternatively, one could estimate the rows of $\mathbf{A}^{-1}$. Algorithm 1 shows how this can be performed by a single server.

Algorithm 1: Estimating $\mathbf{A}^{-1}$

Input: $\mathbf{A} \in \mathrm{GL}_N(\mathbb{R})$
for *i=1 to N* **do**
$\quad$ approximate $\hat{\mathbf{b}}_i \approx \arg \min_{\mathbf{b} \in \mathbb{R}^N} \left\{ \|\mathbf{Ab} - \mathbf{e}_i\|_2^2 \right\}$
end
return $\widehat{\mathbf{A}^{-1}} \leftarrow \begin{bmatrix} \hat{\mathbf{b}}_1 & \cdots & \hat{\mathbf{b}}_N \end{bmatrix}$

In the case where SD is used to approximate $\hat{\mathbf{b}}_i$ from (4), the overall operation count is $\mathcal{O}(\mathcal{T}_i N^2)$; for $\mathcal{T}_i$ the total number of descent iterations used. An upper bound on the number of iterations can be determined by the underlying termination criterion, *e.g.* the criterion $g_i(\hat{\mathbf{b}}^{[t]}) - g_i(\mathbf{b}_{ls}^\star) \leqslant \epsilon$ is guaranteed to be satisfied after $\mathcal{T} = \mathcal{O}(\log(1/\epsilon))$ iterations [36]. The overall error of $\widehat{\mathbf{A}^{-1}}$ may be quantified as

- $\mathrm{err}_{\ell_2}(\widehat{\mathbf{A}^{-1}}) := \|\widehat{\mathbf{A}^{-1}} - \mathbf{A}^{-1}\|_2$
- $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}}) := \|\widehat{\mathbf{A}^{-1}} - \mathbf{A}^{-1}\|_F$

- $\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}}) := \frac{\|\widehat{\mathbf{A}^{-1}} - \mathbf{A}^{-1}\|_F}{\|\mathbf{A}^{-1}\|_F} = \frac{\left(\sum\limits_{i=1}^{N} \|\mathbf{A}\hat{\mathbf{b}}_i - \mathbf{e}_i\|_2^2 \right)^{1/2}}{\|\mathbf{A}^{-1}\|_F}.$

To approximate $\mathbf{A}^{-1}$ distributively, each of the n workers are asked to estimate τ-many $\hat{\mathbf{b}}_i$'s in parallel. When using SD, the worst-case runtime by the workers is $\mathcal{O}(\tau \mathcal{T}_{\max} N^2)$, for $\mathcal{T}_{\max}$ the maximum number of iterations of SD among the workers. If CG is used, each worker needs no more than a total of $N\tau$ CG steps to exactly compute its task, *i.e.* $\mathcal{O}(\tau N \kappa_2)$ operations; as each instance of (4) is expected to converge in $\mathcal{O}(\kappa_2)$ iterations, which is the worst case runtime [37], [38].

In order to bound $\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}}) = \frac{\|\widehat{\mathbf{A}^{-1}} - \mathbf{A}^{-1}\|_F}{\|\mathbf{A}^{-1}\|_F}$, we first upper bound the numerator and then lower bound the denominator. Since $\|\mathbf{A}^{-1} - \widehat{\mathbf{A}^{-1}}\|_F^2 = \sum_{i=1}^{N} \|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2$, bounding the numerator reduces to bounding $\|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2$ for all $i \in \mathbb{N}_N$. This is straightforward

$$\begin{aligned}
\|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2 &\overset{\diamond}{\leqslant} 2 \left(\|\mathbf{A}^{-1}\mathbf{e}_i\|_2^2 + \|\hat{\mathbf{b}}_i\|_2^2 \right) \\
&\overset{\$}{\leqslant} 2 \left(\|\mathbf{A}^{-1}\|_2^2 \cdot \|\mathbf{e}_i\|_2^2 + \|\hat{\mathbf{b}}_i\|_2^2 \right) \\
&= 2 \left(1/\sigma_{\min}(\mathbf{A})^2 + \|\hat{\mathbf{b}}_i\|_2^2 \right)
\end{aligned} \tag{5}$$

where in $\diamond$ we use the fact that $\|\mathbf{u} - \mathbf{v}\|_2^2 \leqslant 2(\|\mathbf{u}\|_2^2 + \|\mathbf{v}\|_2^2)$, and in $\$$ the submultiplicativity of the ℓ_2-norm is invoked. For the denominator, by the definition of the Frobenius norm

$$\|\mathbf{A}^{-1}\|_F^2 = \sum_{i=1}^{N} \frac{1}{\sigma_i(\mathbf{A})^2} \geqslant \frac{N}{\sigma_{\max}(\mathbf{A})^2}. \tag{6}$$

By combining (5) and (6) we get

$$\begin{aligned}
\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}}) &\leqslant \sqrt{2} \left(\frac{N/\sigma_{\min}(\mathbf{A})^2 + \sum_{i=1}^{N} \|\hat{\mathbf{b}}_i\|_2^2}{N/\sigma_{\max}(\mathbf{A})^2} \right)^{1/2} \\
&= \sqrt{2} \left(\kappa_2^2 + \frac{\sigma_{\max}(\mathbf{A})^2}{N} \cdot \sum_{i=1}^{N} \|\hat{\mathbf{b}}_i\|_2^2 \right)^{1/2}.
\end{aligned}$$

This is an additive error bound in terms of the problem's condition number, which also shows a dependency on the estimates $\{\hat{\mathbf{b}}_i\}_{i=1}^{N}$. Propositions 3 and 4 give error bounds when using SD and CG as the subroutine of Algorithm 1 respectively.

Proposition 3. *For* $\mathbf{A} \in \mathrm{GL}_N(\mathbb{R})$, *we have* $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}}) \leqslant \frac{\epsilon \sqrt{N/2}}{\sigma_{min}(\mathbf{A})^2}$ *and* $\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}}) \leqslant \frac{\epsilon \sqrt{N/2}}{\sigma_{\min}(\mathbf{A})}$, *when using SD to solve* (4) *with termination criteria* $\|\nabla g_i(\mathbf{b}^{[t]})\|_2 \leqslant \epsilon$ *for each* i.

Proof. Recall that for a strongly-convex function with strong-convexity parameter μ, we have the following optimization gap [36, Section 9.1.2]

$$g_i(\mathbf{b}) - g_i(\mathbf{b}_{ls}^\star) \leqslant \frac{1}{2\mu} \cdot \|\nabla g_i(\mathbf{b})\|_2^2. \tag{7}$$

For $\mathbf{A} \in \mathrm{GL}_N(\mathbb{R})$ in (4), the constant is $\mu = \sigma_{\min}(\mathbf{A})^2$. By fixing $\epsilon = \sqrt{2\sigma_{\min}(\mathbf{A})^2\eta}$, we have $\eta = \frac{1}{2} \cdot \left(\frac{\epsilon}{\sigma_{\min}(\mathbf{A})}\right)^2$. Thus, by (7) and our termination criterion:

$$\|\nabla g_i(\mathbf{b})\|_2 \leqslant \sqrt{2\sigma_{\min}(\mathbf{A})^2\eta} \quad \Longrightarrow \quad g_i(\mathbf{b}) - g_i(\mathbf{b}_{ls}^{\star}) \leqslant \eta ,$$

so when solving (4) we get

$$g_i(\mathbf{b}) - g_i(\mathbf{b}_{ls}^{\star}) = g_i(\mathbf{b}) - 0 = \|\mathbf{A}\hat{\mathbf{b}}_i - \mathbf{e}_i\|_2^2 ,$$

hence

$$\|\mathbf{A}\hat{\mathbf{b}}_i - \mathbf{e}_i\|_2^2 \leqslant \frac{1}{2} \cdot \left(\frac{\epsilon}{\sigma_{\min}(\mathbf{A})}\right)^2 \tag{8}$$

for all $i \in \mathbb{N}_N$. We want an upper bound for each summand $\|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2$ of the numerator of $\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}})^2$:

$$\begin{aligned}
\|\mathbf{A}^{-1}\mathbf{e}_i - \hat{\mathbf{b}}_i\|_2^2 &= \|\mathbf{A}^{-1}(\mathbf{e}_i - \mathbf{A}\hat{\mathbf{b}}_i)\|_2^2 \\
&\leqslant \|\mathbf{A}^{-1}\|_2^2 \cdot \|\mathbf{e}_i - \mathbf{A}\hat{\mathbf{b}}_i\|_2^2 \\
&\overset{\sharp}{\leqslant} \|\mathbf{A}^{-1}\|_2^2 \cdot \frac{1}{2} \cdot \left(\frac{\epsilon}{\sigma_{\min}(\mathbf{A})}\right)^2 \tag{9} \\
&= \frac{\epsilon^2}{2\sigma_{\min}(\mathbf{A})^4} \tag{10}
\end{aligned}$$

where $\sharp$ follows from (8), thus $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}})^2 \leqslant \frac{N\epsilon^2}{2\sigma_{\min}(\mathbf{A})^4}$. Substituting (9) into the definition of $\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}})$ gives us

$$\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}})^2 \leqslant \frac{\|\mathbf{A}^{-1}\|_2^2}{\|\mathbf{A}^{-1}\|_F^2} \cdot \frac{N}{2} \cdot \left(\frac{\epsilon}{\sigma_{\min}(\mathbf{A})}\right)^2 \overset{\ddagger}{\leqslant} \frac{N\epsilon^2/2}{\sigma_{\min}(\mathbf{A})^2}$$

where $\ddagger$ follows from the fact that $\|\mathbf{A}^{-1}\|_2^2 \leqslant \|\mathbf{A}^{-1}\|_F^2$. $\blacksquare$

In the experiments of Subsection IV-A, we verify that Proposition 3 holds for Gaussian random matrices. The dependence on $1/\sigma_{\min}(\mathbf{A})$ is an artifact of using gradient methods to solve the underlying problems (4), since the error will be multiplied by $\|\mathbf{A}^{-1}\|_2^2$. In theory, this can be annihilated if one runs the algorithm on $p\mathbf{A}$ for $p \approx 1/\sigma_{\min}(\mathbf{A})$, followed by multiplication of the final result by p. This is a way of preconditioning SD. In practice, the scalar p should not be selected to be much larger than $1/\sigma_{\min}(\mathbf{A})$, as it could result in $\widehat{\mathbf{A}^{-1}} \approx \mathbf{0}_{N \times N}$.

Proposition 4. *Assume Algorithm 1 uses CG to solve (4). Then, in $\mathcal{O}\left(N\sqrt{\kappa_2}\ln(1/\epsilon)\right)$ iterations, we have $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}}) \leqslant N\epsilon$. Moreover, if $\mathbf{A}^{\top}\mathbf{A}$ has $\tilde{N}$ distinct eigenvalues, it converges in at most $\tilde{N}N$ steps.*

Proof. By [37, Section 10] and [39, Section 2], we know that for each subroutine (4) of Algorithm 3, CG requires at most $\mathcal{O}(\sqrt{\kappa_2}\ln(1/\epsilon))$ iterations in order to attain an ϵ-optimal point, for each $\hat{\mathbf{b}}_i$. Hence, considering all approximate columns $\{\hat{\mathbf{b}}_i\}_{i=1}^N$, we conclude that the total error in terms of the Frobenius norm of $\widehat{\mathbf{A}^{-1}}$, is at most $N\epsilon$.

Recall that in order to solve (1) with the CG method in the case where $\mathbf{A}$ is neither symmetric, positive-definite, nor square, we apply the CG iteration to the normal equations: $\mathbf{A}^{\top}\mathbf{A}\mathbf{y} = \mathbf{A}^{\top}\theta$. This follows by setting the derivative of (1) to zero. In our scenario, we are assuming that $\mathbf{A} \in \mathrm{GL}_N(\mathbb{R})$, hence $\mathbf{A}^{\top}\mathbf{A}$ is full-rank and symmetric, thus CG in its simplest form can be used to solve the minimization problems of Algorithm 1. By [38, Theorem 38.4], it follows that each instance of (4) converges in at most $\tilde{N}$ steps. $\blacksquare$

Even though Proposition 4 guarantees convergence in at most $\tilde{N}N$ steps, it does not assume finite floating-point precision. Therefore, this does not hold in practical settings. Our experiments though show that after significantly less steps, we achieve approximations of negligible error, which is sufficient for ML and FL applications.

A. Numerical Experiments

The accuracy of the proposed algorithm was tested on randomly generated matrices, using both SD and CG for the subroutine optimization problems. The depicted results are averages of 20 runs, with termination criteria $\|\nabla g_i(\mathbf{b}^{[t]})\|_2 \leqslant \epsilon$ for SD and $\|\mathbf{b}_i^{[t]} - \mathbf{b}_i^{[t-1]}\|_2 \leqslant \epsilon$ for CG, for the given ϵ accuracy parameters. We considered $\mathbf{A} \in \mathbb{R}^{100 \times 100}$. The error subscripts represent $\mathscr{A} = \{\ell_2, F, \mathrm{r}F\}$, $\mathscr{N} = \{\ell_2, F\}$. We note that significantly fewer iterations took place when CG was used for the same ϵ, though this depends heavily on the choice of the step-size. The errors observed in the case of CG, are due to floating-point arithmetic. Therefore, as expected; there is a trade-off between accuracy and speed when using SD vs. CG.

Average $\widehat{\mathbf{A}^{-1}}$ errors, for $\mathbf{A} \sim 50 \cdot \mathcal{N}(0,1)$ — SD					
ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
$\mathrm{err}_{\mathscr{A}}$	$\mathcal{O}(10^{-2})$	$\mathcal{O}(10^{-5})$	$\mathcal{O}(10^{-7})$	$\mathcal{O}(10^{-9})$	$\mathcal{O}(10^{-12})$

Average $\widehat{\mathbf{A}^{-1}}$ errors, for $\mathbf{A} \sim 50 \cdot \mathcal{N}(0,1)$ — CG					
ϵ	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$\mathrm{err}_{\mathcal{N}}$	$\mathcal{O}(10^{-3})$	$\mathcal{O}(10^{-5})$	$\mathcal{O}(10^{-8})$	$\mathcal{O}(10^{-11})$	$\mathcal{O}(10^{-12})$
$\mathrm{err}_{\mathrm{r}F}$	$\mathcal{O}(10^{-3})$	$\mathcal{O}(10^{-5})$	$\mathcal{O}(10^{-7})$	$\mathcal{O}(10^{-10})$	$\mathcal{O}(10^{-12})$

We utilized Algorithm 1 in Newton's method, for classifying images of four and nine from MNIST, by solving a regularized logistic regression minimization problem. For Algorithm 1, we used CG with a fixed number of iteration per column estimation. It is clear from Figure 1 that we require no more than 18 iterations per column estimate, for $N = 785$, to attain the optimal classification rate. With more than 18 CG iterations, the same classification rate was obtained.

Fig. 1. MNIST classification error, where Algorithm 1 is used in Newton's method. In red, we depict the error when exact inversion was used.

V. SECURE CODED MATRIX INVERSION

In this section, we describe the proposed CMIM (also presented in [9]) which makes Algorithm 1 resilient to stragglers, and show how it can be applied to the relaxed FL scenario described in the introduction. The CMIM workflow is depicted in Figure 2.

Our scheme can be broken up in to four phases: (a) the coordinator shares elements $\beta, \mathcal{H}$ of a finite field with all the clients, (b) the clients each generate a *pseudorandom permutation* (PRP) σ_ι, encrypt their corresponding data block $\mathbf{A}_\iota$ through a matrix polynomial $f_\iota(\mathsf{x})$, and broadcast $\{f_\iota(\mathsf{x}), \sigma_\iota\}$ to the other clients, (c) the clients recover $\mathbf{A}$, compute and encode their assigned task $\mathbf{W}_\iota$, which is communicated to the coordinator, (d) the coordinator decodes once sufficiently many workers respond. It is also possible that $\beta, \mathcal{H}$ are determined collectively by the clients, or by a single client, which makes the data sharing secure against a curious and dishonest coordinator.

Fig. 2. Algorithmic workflow of the CMIM, as proposed in [9]. The master shares $f(\mathsf{x})$, an encoding analogous to (12), along with $\beta, \{\eta_j^{-1}\}_{j=1}^k$. The workers then recover $\mathbf{A}$, compute their assigned tasks, and encode them according to $\mathbf{G}$. Once k encodings $\mathbf{W}_\iota$ are sent back, $\widehat{\mathbf{A}^{-1}}$ can be recovered.

In our proposed approach, we assume there is a trustworthy coordinator who shares certain parameters to each of the k clients which constitute the network; *e.g.* hospitals in a health care network, each of which are comprised of multiple servers. What we present works for the case where the clients have local datasets of different sizes, $\{N_i\}_{i=1}^k$. This would result in the encoding functions $f_\iota(\mathsf{x})$ having different degrees, or their matrix coefficients being of a different size. In our setting we assume the workers are *homogeneous*, *i.e.* they have the same computational power. Therefore, equal computational loads are assigned to each of them. In order to keep the notation and size of the communication loads consistent, we assume w.l.o.g. that $\mathbf{A}_\iota \in \mathbb{R}^{N \times T}$ for all $\iota \in \mathbb{N}_k$. If this is not the case, before $f_\iota(\mathsf{x})$ are determined, the clients could perform a data exchange phase (*e.g.* [15]), so that $N_i = N_j$ for all $i \neq j$. By this, it follows that the number of blocks does not have to be equal to the number of clients. The example we describe, is simply a motivation. A flowchart of our approach is presented in Figure 3.

Moreover, in the case where $M > N$; for $M = \sum_{i=1}^k N_i$, we can select a subset of features and/or samples, so that the resulting data matrix we consider is square. This can be interpreted as using the surrogate $\tilde{\mathbf{A}} = \mathbf{S}\mathbf{A}$, where $\mathbf{S} \in \mathbb{R}^{N \times M}$ is an appropriate (sparse) *sketching* matrix for matrix inversion [40], which the workers agree on.

First, in V-A we argue why all of $\mathbf{A}$ needs to be known by each of the workers, in order to recover entries or columns of its inverse. Then, in V-B we focus on phases (a) and (b), where we utilize Lagrange interpolation to securely share $\mathbf{A}$ among the

Fig. 3. Flowchart of our proposal, where $k = n_i = 4$ for all $i \in \mathbb{N}_4$.

workers. We discuss the computation tasks the workers are requested to compute, which are blocks of $\widehat{\mathbf{A}^{-1}}$; and collectively correspond to the subroutine problems of Algorithm 1. In V-C we focus on (c) and (d), where we show how the workers encode their computations, and describe the coordinator's decoding step. Optimality of BRS generator matrices in terms of the encoded communication loads is established in V-D.

When assuming no floating-point errors, our approach introduces no numerical nor approximation errors. The errors are a consequence of using iterative solvers to estimate (4), which we utilize to linearly separate the computations. Therefore, if the workers can recover the optimal solutions to the underlying minimization problems, our scheme would be *exact*.

A. Knowledge of $\mathbf{A}$ is necessary

A bottleneck when computing the inverse of a matrix; or estimating its columns, is that the entire matrix needs to be known. A single change in the matrix's entries may result in a non-singular matrix, which conveys how sensitive Gaussian elimination is. Such problems are extensively studied in conditioning and stability of numerical analysis [38], and in perturbation theory. This is not a focus of our work.

In the case where only one column is not known, one can determine the subspace in which the missing column lies, but without the knowledge of at least one entry of that column, it would be impossible to recover that column. Even with such an approach or a matrix completion algorithm, the entire $\mathbf{A}$ is determined before we proceed to inverting $\mathbf{A}$; or performing linear regression to approximate $\mathbf{Ab} = \mathbf{e}_i$ as in (4).

A similar issue, relating to our set up, is the case where one of the blocks is different. This could lead to drastic miscalculations. In the following example, we consider $n = k = 2$ and $N = 4$, where the second worker sends two different blocks, which are indicated by a different color and font:

$$A_1 = \begin{pmatrix} 6 & 2 & 2 & \text{-5} \\ 0 & -1 & 2 & 0 \\ -5 & 6 & \text{-1} & \text{-3} \\ 5 & -3 & \text{-4} & 3 \end{pmatrix} \quad A_2 = \begin{pmatrix} 6 & 2 & -1 & -3 \\ 0 & -1 & 5 & 6 \\ -5 & 6 & 3 & -2 \\ 5 & -3 & 1 & 6 \end{pmatrix}.$$

It follows that $\|A_1^{-1}\|_F \approx 90.45$, $\|A_2^{-1}\|_F \approx 1$, and $\|A_1^{-1} - A_2^{-1}\|_0 = 16$; *i.e.* no entries of A_1^{-1} and A_2^{-1} are equal.

Furthermore, by the data processing inequality [41, Corollary pg. 35], the above imply that no less than N^2 information symbols can be known by each worker, while hoping to approximate a column of $\mathbf{A}^{-1}$. Hence, all clients need full knowledge of each others information, and cannot communicate less than NT symbols to each other. This is a consequence of the fact that a dense vector is not recoverable from underdetermined linear measurements. They can however send an encoded version of their respective block $\mathbf{A}_\iota \in \mathbb{R}^{N \times T}$ to the other clients consisting of NT symbols, determined by a modified Lagrange polynomial, which guarantees security against eavesdroppers.

Similar cryptographic protocols date back to the SSS algorithm [42], [43], which is also based on RS codes. This idea has been extensively exploited in LCC [4], yet differs from our approach.

B. Phases $(a), (b)$ — Data Encryption and Sharing

Let $k, \gamma \in \mathbb{Z}_+$ be factors of N and T respectively, so that $T = \frac{N}{k}$ and $\Gamma = \frac{T}{\gamma}$.[3] The coordinator constructs a set of distinct *interpolation points* $\mathcal{B} = \{\beta_j\}_{j=1}^n \subsetneq \mathbb{F}_q^\times$, for $q > n \geqslant \gamma$.[4] To construct this set, it suffices to sample $\beta \in \mathbb{F}_q^\times$; any one of the $\phi(q-1)$ primitive roots of $\mathbb{F}_q$ (ϕ is Euler's totient function), which is a generator of the multiplicative group $(\mathbb{F}_q^\times, \cdot)$, and define each point as $\beta_j = \beta^j$. Then, a random multiset $\mathcal{H} = \{\eta_j \in \mathbb{F}_q^\times \mid \forall j \in \mathbb{N}_\gamma\}$ of size γ is generated, *i.e.* repetitions in $\mathcal{H}$ are allowed, which will be used to remove the structure of the Lagrange coefficients, as the adversaries could exploit their structure to reveal β.

The element β and set $\mathcal{H}$, are broadcasted securely to all the workers through a public-key cryptosystem, *e.g.* RSA or McEliece. Matrices $\mathbf{A}_\iota$ are partitioned into γ blocks

$$\mathbf{A}_\iota = \begin{bmatrix} \mathbf{A}_\iota^1 & \cdots & \mathbf{A}_\iota^\gamma \end{bmatrix} \quad \text{where } \mathbf{A}_\iota^i \in \mathbb{R}^{N \times \Gamma}, \; \forall i \in \mathbb{N}_\gamma, \tag{11}$$

and each client generates a PRP $\sigma_\iota \in S_\gamma$. The blocks $\{\mathbf{A}_\iota\}_{\iota=1}^k$ are encrypted locally through the univariate polynomials

$$f_\iota(\mathsf{x}) = \sum_{j=1}^\gamma \mathbf{A}_\iota^j \cdot \eta_{\sigma_\iota(j)} \left(\prod_{l \neq j} \frac{\mathsf{x} - \beta_l}{\beta_j - \beta_l} \right) \tag{12}$$

for which $f_\iota(\beta_j) = \eta_{\sigma_\iota(j)} \mathbf{A}_\iota^j$.

The clients securely broadcast $\{f_\iota(\mathsf{x}), \sigma_\iota\}$ to each other, and their servers can then recover all $\mathbf{A}_\iota$'s as follows:

$$\mathbf{A}_\iota = \begin{bmatrix} \eta_{\sigma_\iota(1)}^{-1} f_\iota(\beta_1) & \cdots & \eta_{\sigma_\iota(\gamma)}^{-1} f_\iota(\beta_\gamma) \end{bmatrix} \in \mathbb{R}^{N \times T}. \tag{13}$$

The coefficients of $f_\iota(\mathsf{x})$ are comprised of $N\Gamma$ symbols, thus, each polynomial consists of a total of NT symbols, which is the minimum number of symbols needed to be communicated. The PRP σ_ι is generated locally by the clients, to ensure that each $f_\iota(\mathsf{x})$ differs by more than just the matrix partitions.

We assume Kerckhoffs' principle, which states that everyone has knowledge of the system, including the messages $f_\iota(\mathsf{x})$. For the proposed CMIM, as long as $\{\beta, \mathcal{H}\}$ and σ_ι are securely communicated, even if $f_\iota(\mathsf{x})$ is revealed, the block $\mathbf{A}_\iota$ is secure against polynomial-bounded adversaries (this is the security level assumed by the cryptosystems used for the communication).

Proposition 5. *The encryptions of $\mathbf{A}_\iota$ through $f_\iota(\mathsf{x})$, are as secure against eavesdroppers as the public-key cryptosystems which are used when broadcasting $\{\beta, \mathcal{H}\}$ and σ_ι. To recover $\mathbf{A}_\iota$, an adversary needs to intercept both communications, and break both cryptosystems.*

Proof. We prove this by contradiction. Assume that an adversary was able to reverse the encoding $f_\iota(\mathsf{x})$ of $\mathbf{A}_\iota$. This implies that he was able to reveal β and $\sigma_\iota(\mathcal{H}) \coloneqq \{\eta_{\sigma_\iota(j)}\}_{j=1}^\gamma$. The only way to reveal these elements, is if he was able to both intercept and decipher the public-key cryptosystem used by the coordinator, which contradicts the security of the cryptosystem.

In order to invert the multiplications of $\sigma_\iota(\mathcal{H})$ for each of the evaluations of $f_\iota(\mathsf{x})$, both $\mathcal{H}$ and σ_ι need to be known. To do so, the adversary needs to intercept both the communication between the coordinator and the clients, and the communication between the clients, as well as breaking both the cryptosystems used to securely carry out these communications. ∎

C. Phases $(c), (d)$ — Computations, Encoding and Decoding

At this stage, the workers have knowledge of everything they need in order to recover $\mathbf{A}$, before they carry out their computation tasks. By (13), the recovery is straightforward.

For Algorithm 1, any CCM in which the workers compute an encoding of partitions of the resulting computation $\mathbf{E} = \begin{bmatrix} E_1 & \cdots & E_k \end{bmatrix}$ could be utilized. It is crucial that the encoding takes place on the computed tasks $\{E_i\}_{i=1}^k$ in the scheme, and *not* the assigned data or partitions of the matrices that are being computed over (such CMM leverage the linearity of matrix multiplication), otherwise the algorithm could potentially not return the correct approximation. This also means that utilizing such encryption approaches (*e.g.* [4]) for guaranteeing security against the workers, is not an option. We face these restrictions due to the fact that matrix inversion is a non-linear operator.

The computation tasks E_i correspond to a partitioning $\widehat{\mathbf{A}^{-1}} = \begin{bmatrix} \hat{\mathcal{A}}_1 & \cdots & \hat{\mathcal{A}}_k \end{bmatrix}$, of our approximation from Algorithm 1. We propose a linear encoding of the computed blocks $\{\hat{\mathcal{A}}_i\}_{i=1}^k$ based on generators satisfying (2). Along with the proposed decoding step, we have a MDS-based CCM for matrix inversion.

We consider the same parameters as in V-B, in order to reuse $\mathcal{B}$ in the proposed CMIM. Each $\hat{\mathcal{A}}_i$ is comprised of T distinct but consecutive approximations of (4), *i.e.*

$$\hat{\mathcal{A}}_i = \begin{bmatrix} \hat{\mathbf{b}}_{(i-1)T+1} & \cdots & \hat{\mathbf{b}}_{iT} \end{bmatrix} \in \mathbb{R}^{N \times T} \quad \forall i \in \mathbb{N}_k,$$

---

[3]If $\gamma \nmid T$, append $\mathbf{0}_{T \times 1}$ to the end of the first $\tilde{\gamma} = T(\mathrm{mod}\,\gamma)$ blocks which are each comprised of $\tilde{\Gamma} = \lfloor \frac{T}{\gamma} \rfloor$ columns of $\mathbf{A}_\iota$, while the remaining $\gamma - \tilde{\gamma}$ blocks are comprised of $\tilde{\Gamma} + 1$ columns. Now, each block is of size $T \times (\tilde{\Gamma} + 1)$.

[4]For the encodings of the $\mathbf{A}_\iota$'s, γ points suffice, and we only need to require $q > \gamma$. We select $\mathcal{B}$ of cardinality n and require $q > n \geqslant \gamma$, in order to reuse $\mathcal{B}$ in our CCM.

which could also be approximated by iteratively solving

$$\hat{\mathcal{A}}_i \approx \arg\min_{\mathbf{B} \in \mathbb{R}^{N \times T}} \left\{ \left\| \mathbf{A}\mathbf{B} - \left[\mathbf{e}_{(i-1)T+1} \cdots \mathbf{e}_{iT} \right] \right\|_F^2 \right\}.$$

Without loss of generality, we assume that the workers use the same algorithms and parameters for estimating the columns $\{\hat{\mathbf{b}}_i\}_{i=1}^N$. Therefore, workers allocated the same tasks are expected to get equal approximations in the same amount of time.

For our CCM, we leverage BRS generator matrices for both the encoding and decoding steps. We adapt the GC framework, so we need an analogous condition to $\mathbf{a}_{\mathcal{I}}^\top \mathbf{G}_{\mathcal{I}} = \vec{1}$ for the CMIM; in order to invoke Algorithm 1. The condition we require is $\tilde{\mathbf{D}}_{\mathcal{I}}\tilde{\mathbf{G}}_{\mathcal{I}} = \mathbf{I}_N$, for an encoding-decoding pair $(\tilde{\mathbf{G}}, \tilde{\mathbf{D}}_{\mathcal{I}})$.

From our discussion on BRS codes in III-A, we set $\tilde{\mathbf{G}} = \mathbf{I}_T \otimes \mathbf{G}$ and $\tilde{\mathbf{D}}_{\mathcal{I}} = \mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}^{-1}$ for any given set of k responsive workers indexed by $\mathcal{I}$. The index set of blocks requested from the ιth worker to compute is $\tilde{\mathcal{J}}_\iota := \text{supp}(\mathbf{G}_{(\iota)})$, and has cardinality w. The workers' encoding steps correspond to

$$\tilde{\mathbf{G}} \cdot (\widehat{\mathbf{A}^{-1}})^\top = (\mathbf{I}_T \otimes \mathbf{G}) \cdot \begin{bmatrix} \hat{\mathcal{A}}_1^\top \\ \vdots \\ \hat{\mathcal{A}}_k^\top \end{bmatrix} = \begin{pmatrix} \sum_{j \in \mathcal{J}_1} p_j(\beta_1) \cdot \hat{\mathcal{A}}_j^\top \\ \vdots \\ \sum_{j \in \mathcal{J}_n} p_j(\beta_n) \cdot \hat{\mathcal{A}}_j^\top \end{pmatrix} \tag{14}$$

which are carried out locally, once they have computed their assigned tasks. We denote the encoding of the ιth worker by $\mathbf{W}_\iota \in \mathbb{C}^{T \times N}$, i.e. $\mathbf{W}_\iota = \sum_{j \in \mathcal{J}_\iota} p_j(\beta_\iota) \cdot \hat{\mathcal{A}}_j^\top$, which is sent to the coordinator. The received encoded computations by any distinct k workers indexed by $\mathcal{I}$, constitute $\tilde{\mathbf{G}}_{\mathcal{I}} \cdot (\widehat{\mathbf{A}^{-1}})^\top$.

Lemma 1 implies that as long as k workers respond, the approximation $\widehat{\mathbf{A}^{-1}}$ is recoverable. Moreover, the decoding step reduces to a matrix multiplication of $k \times k$ matrices. Applying $\mathbf{H}_{\mathcal{I}}^{-1}$ to a square matrix can be done in $\mathcal{O}(k^2 \log k)$, through the IFFT algorithm. The prevailing computation in our decoding, is applying $\mathbf{P}^{-1}$. The decoding step is

$$\begin{aligned} \tilde{\mathbf{D}}_{\mathcal{I}} \cdot \left(\tilde{\mathbf{G}}_{\mathcal{I}} \cdot (\widehat{\mathbf{A}^{-1}})^\top \right) &= \left(\mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}^{-1} \right) \cdot \left(\mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}} \right) \cdot (\widehat{\mathbf{A}^{-1}})^\top \\ &= (\mathbf{I}_T \cdot \mathbf{I}_T) \otimes (\mathbf{G}_{\mathcal{I}}^{-1} \cdot \mathbf{G}_{\mathcal{I}}) \cdot (\widehat{\mathbf{A}^{-1}})^\top \\ &= \mathbf{I}_T \otimes \mathbf{I}_k \cdot (\widehat{\mathbf{A}^{-1}})^\top \\ &= (\widehat{\mathbf{A}^{-1}})^\top \end{aligned}$$

and our scheme is valid.

The above CCM therefore has a linear encoding done locally by the workers (14), is MDS since $s = d - 1$, and its decoding step reduces to computing and applying $\mathbf{G}_{\mathcal{I}}^{-1}$ (Lemma 1). The security of the encodings rely on the secrecy of $\mathcal{B}$, which were sent from the coordinator to the workers. For an additional security layer, the interpolation points of $\mathcal{B}$ could instead be defined as $\beta_j = \beta^{\pi(j)}$, for $\pi \in S_n$ a PRP. In this case, π^{-1} would also need to be securely broadcasted.

Fig. 4. Comparison of decoding complexity, when naive matrix inversion is used (so $\mathcal{O}(k^3)$) compared to the decoding step implied by Lemma 1, for $n = 200$ and varying s. We also provide a logarithmic scale comparison.

Remark 6. *With the above framework, any sparsest-balanced generator MDS matrix [31] would suffice, as long as it satisfies the MDS theorem [44]. By Lemma 1, if we set $k = \Omega(\sqrt{N})$ (as in [8]), the decoding step could then be done in $\mathcal{O}(N^{\omega/2}) = o(N^{1.186})$ time, which is close to linear in terms of N.*

Theorem 7. *Let $\mathbf{G} \in \mathbb{F}^{n \times k}$ be a generator matrix of any MDS code over $\mathbb{F}$, for which $\|\mathbf{G}^{(j)}\|_0 = n - k + 1$ and $\|\mathbf{G}_{(i)}\|_0 = w$ for all $(i, j) \in \mathbb{N}_n \times \mathbb{N}_k$. By utilizing Algorithm 1, we can devise a linear MDS coded matrix inversion scheme; through the encoding-decoding pair $(\tilde{\mathbf{G}}, \tilde{\mathbf{D}}_{\mathcal{I}})$.*

Proof. The encoding coefficients applied locally by each of the n workers correspond to a row of $\mathbf{G}$. The encodings of all the workers then correspond to $\tilde{\mathbf{G}} \cdot (\widehat{\mathbf{A}^{-1}})^\top$, for $\tilde{\mathbf{G}} = \mathbf{I}_T \otimes \mathbf{G}$, as in (14). Consider any set of responsive workers $\mathcal{I}$ of size k, whose encodings constitute $\tilde{\mathbf{G}}_{\mathcal{I}} \cdot (\widehat{\mathbf{A}^{-1}})^\top$. By the MDS theorem, $\mathbf{G}_{\mathcal{I}}$ is invertible. Hence, the decoding step reduces to inverting $\mathbf{G}_{\mathcal{I}}$; i.e. $\tilde{\mathbf{D}}_{\mathcal{I}} = \mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}^{-1}$, and is performed online. $\blacksquare$

Constructions based on cyclic MDS codes, which have been used to devise GC schemes [45], can also be considered. These encoding matrices are not sparsest-balanced, which makes them suitable when considering *heterogeneous* workers.

Proposition 8. *Any cyclic $[n, k]$ MDS code $\mathcal{C}$ over $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ can be used to devise a coded matrix inversion encoding-decoding pair $(\tilde{\mathbf{G}}, \tilde{\mathbf{D}}_{\mathcal{I}})$.*

Proof. Consider a cyclic $[n, n - s]$ MDS code $\mathcal{C}$ over $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$. Recall that from our assumptions, we have $s = n - k$. By [45, Lemma 8], there exists a codeword $\mathbf{g}_1 \in \mathcal{C}$ of support $d = s + 1$, *i.e.* $\|\mathbf{g}_1\|_0 = d$. Since $\mathcal{C}$ is cyclic, it follows that the cyclic shifts of $\mathbf{g}_1$ also lie in $\mathcal{C}$. Denote the $n - 1$ consecutive cyclic shifts of $\mathbf{g}_1$ by $\{\mathbf{g}_i\}_{i=2}^n \subsetneq \mathcal{C} \subsetneq \mathbb{F}^{1 \times n}$, which are all distinct. Define the cyclic matrix

$$\bar{\mathbf{G}} := \begin{pmatrix} | & | & & | \\ \mathbf{g}_1^\top & \mathbf{g}_2^\top & \cdots & \mathbf{g}_n^\top \\ | & | & & | \end{pmatrix} \in \mathbb{F}^{n \times n}.$$

Since $\|\mathbf{g}_i\|_0 = d$ and $\mathbf{g}_i$ is a cyclic shift of $\mathbf{g}_{i-1}$ for all $i > 1$, it follows that $\|\bar{\mathbf{G}}_{(i)}\|_0 = \|\bar{\mathbf{G}}_{(j)}\|_0 = d$ for all $i, j \in \mathbb{N}_n$, *i.e.* $\bar{\mathbf{G}}$ is sparsest and balanced. If we erase *any* $s = n - k$ columns of $\bar{\mathbf{G}}$, we get $\mathbf{G} \in \mathbb{F}^{n \times k}$. By erasing arbitrary columns of $\bar{\mathbf{G}}$, the resulting $\mathbf{G}$ is *not* balanced, *i.e.* we have $\|\mathbf{G}_{(i)}\|_0 \neq \|\mathbf{G}_{(j)}\|_0$ for some pairs $i, j \in \mathbb{N}_n$. Similar to our construction based on BRS generator matrices, we define the encoding matrix to be $\tilde{\mathbf{G}} = \mathbf{I}_T \otimes \mathbf{G}$. The local encodings are then analogous to (14).

Consider an arbitrary set of k non-straggling workers $\mathcal{I} \subsetneq \mathbb{N}_n$, and the corresponding matrix $\mathbf{G}_{\mathcal{I}} \in \mathbb{F}^{k \times k}$. By [45, Lemma 12, B4.], $\mathbf{G}_{\mathcal{I}}$ is invertible. The decoding matrix is then $\tilde{\mathbf{D}}_{\mathcal{I}} = \mathbf{I}_T \otimes \mathbf{G}_{\mathcal{I}}^{-1}$, and the condition $\tilde{\mathbf{D}}_{\mathcal{I}} \tilde{\mathbf{G}} = \mathbf{I}_N$ is met. ∎

D. Optimality of MDS BRS Codes

Under the assumption that $k = n - s$, by utilizing the $\mathsf{BRS}_q[n, k]$ generator matrices, we achieved the minimum possible communication load from the workers to the coordinator. From our discussion in V-A, we cannot hope to receive an encoding of less than N^2/k symbols; when we require that k workers respond with the same amount of information symbols in order to recover $\widehat{\mathbf{A}^{-1}} \in \mathbb{R}^{N \times N}$, unless we make further assumptions on the structure of $\mathbf{A}$ and $\mathbf{A}^{-1}$. Each encoding $\mathbf{W}_\iota$ consists of $NT = N^2/k$ symbols, so we have achieved the lower bound on the minimum amount of information needed to be sent to the coordinator. Moreover, $\mathbf{W}_\iota \in \mathbb{C}^{T \times N}$ for any sparsest-balance generator MDS matrix. This also holds true for other generator matrices which can be used in Theorem 7, as the encodings are linear (*e.g.* Proposition 8).

We also require the workers to estimate the least possible number of columns for the given recovery threshold k. For our choice of parameters, the bound of [22, Theorem 1] is met with equality. That is, for all $i \in \mathbb{N}_n$:

$$\|\mathbf{G}_{(i)}\|_0 = w = \frac{k}{n} \cdot d = \frac{k}{n} \cdot (n - k + 1),$$

which means that for homogeneous workers, we cannot get a sparser generator matrix. This, along with the requirement that $\mathbf{G}_{\mathcal{I}}$ should be invertible for all possible $\mathcal{I}$, are what we considered in (2).

E. Time and Space Complexity

Next, we discuss the complexity of our method. Communication loads and storage are measured in symbols over $\mathbb{R}$. For simplicity, we assume that the n workers are homogeneous and the local data blocks $\{\mathbf{A}_\iota\}_{\iota=1}^k$ are of size $N \times T$, for $T = N/k = \Gamma\gamma$. To further simplify our expressions and for fair comparisons to other polynomial codes, we set $k = n = \Omega(\sqrt{N})$ (as in [8]).

Let $\nu = |\mathsf{Enc}(\beta)|$ denote the number of symbols required for the encoding of β through a public-key cryptosystem used to securely broadcast a symbol. Further note that $|\mathsf{Enc}(\mathcal{H})| = \gamma\nu$ and $|\mathsf{Enc}(\sigma_\iota)| \leqslant \gamma\nu/2$, when the same cryptosystem is used to broadcast $\mathcal{H}$ and σ_ι respectively. Hence, phase (a) requires a communication load of $\mathcal{O}(\nu\gamma)$ symbols per client.

There are γ modified Lagrange polynomials in (12); each of which require $\mathcal{O}(n - 1)$ operations to compute. Multiplying each polynomial with one of the γ sub-blocks $\{\mathbf{A}_\iota^j\}_{j=1}^\gamma$, requires $\gamma \cdot \mathcal{O}(N\Gamma) = \mathcal{O}(NT) = \mathcal{O}(N^{3/2})$ additional operations in total. Finally, summing over the encoded blocks requires $\mathcal{O}(N\Gamma(\gamma - 1)) = \mathcal{O}(N^{3/2})$ operations. Hence, the encoding through the polynomials $f_\iota(\mathsf{x})$ has complexity $\mathcal{O}(2N^{3/2} + \gamma(n - 1)) = \mathcal{O}(\sqrt{N}(N + \gamma))$ for each data block, which is done locally by the clients. The PRP used could be any block cipher, *e.g.* the Feistel cipher; which has time complexity $\mathcal{O}(n)$. Therefore, phase (b) has complexity $\mathcal{O}(\sqrt{N}(N + \gamma + 1))$ and the clients communicate $2NT + n = \Omega(\sqrt{N}(N + 1))$ symbols to each other, which accounts for a total communication load of $(k - 1) \cdot \Omega(\sqrt{N}(N + 1)) = \Omega(N^2)$ symbols per client.

At phase (c), the ιth client first recovers $\mathbf{A}$ by evaluating $\eta_{\sigma_\iota(j)}^{-1} f_\iota(\beta_j)$; for each $j \in \mathbb{N}_\gamma$, which requires a total of $\mathcal{O}(\gamma \cdot N\Gamma) = \mathcal{O}(N^{3/2})$ operations. The complexity of the computation tasks depends on the underlying optimization algorithm used by the workers; and the desired level of accuracy. By Proposition 4, under the given assumptions, when using CG we converge after $\tilde{N}$ iterations per column estimate, and each iteration has complexity $\mathcal{O}(N)$. Therefore, the complexity of the workers' tasks are $\mathcal{O}(\tilde{N}NT) = \mathcal{O}(\tilde{N}N^{3/2})$. All in all, the computation tasks at phase (c) have total complexity $\mathcal{O}((\tilde{N} + 1)N^{3/2}) = \mathcal{O}(\tilde{N}N^{3/2})$ per worker. Since $\mathbf{W}_\iota \in \mathbb{C}^{T \times N}$ for each ι, the communication load is $2NT = 2N^2/k = \Omega(N^{3/2})$ symbols. Furthermore, the baseline to computing the $\mathbf{A}^{-1}$ is Gaussian elimination; which has complexity $\mathcal{O}(N^3)$ when carried out on one server, while our approach through CG has complexity $k \cdot \mathcal{O}(\tilde{N}N^{3/2}) = \mathcal{O}(\tilde{N}N^2)$.

By Lemma 1 and Remark 6; the decoding takes $\mathcal{O}(N^{\omega/2}) = o(N^{1.186})$ time, which amounts to the complexity of phase (d). Since our recovery threshold is k, phase (d) no more than $k \cdot \Omega(NT) = \Omega(N^2)$ symbols need to be received and stored by the coordinator, who finally recovers a matrix of size $N \times N$.

We summarize the communication loads (C.L.) and time complexity (T.C.) of each of the four phases in the Table below. The time complexity for phase (a) depends on the encryption method that is used to securely communicate $\beta, \mathcal{H}$; which we do not study, and there is no communication taking place in phase (d). Phases (a) and (d) correspond to the coordinator, (b) to each client, and (c) to each worker.

Communication Loads & Time Complexities				
Phase	(a) *share* $\beta, \mathcal{H}$	(b) *encrypt* $\mathbf{A}_\iota$	(c) *CC job*	(d) *decode*
C.L.	$\mathcal{O}(\nu\gamma)$	$\Omega(N^2)$	$\Omega(N^{3/2})$	✗
T.C.	✗	$\mathcal{O}(\sqrt{N}(N+\gamma))$	$\mathcal{O}(\tilde{N}N^{3/2})$	$o(N^{1.186})$

A bottleneck of our approach is the workers' storage requirement. As was discussed in V-A, the workers need to recover $\mathbf{A}$, so they need to store a total of N^2 symbols. The central server receives a total of k completed tasks $\{\mathbf{W}_i\}_{i\in\mathcal{I}}$, which constitute to a total of $2N^2$ symbols. Further examining this drawback would be worthwhile future work.

F. Comparison to Exact Matrix Inversion

We conclude this section with a discussion on the conditions under which our CMIM will have advantages over standard matrix inversion approaches. First of all, the main bottleneck of our approach is the fact that each worker has a storage requirement of N^2 symbols. When N is relatively small, matrix inversion can be performed by a single server; though the time complexity is still high, in which case our distributed approach is beneficial. In this scenario, the storage constraint is not an issue. For N very large, our approach is still advantageous in terms of time complexity, as a single server would need to perform the entire computation on its own; while also storing the entire matrix. In this case, the storage requirement of our approach is disadvantageous, since we require total storage of kN^2 symbols across the network, while matrix inversion only requires N^2. This is the cost we pay for performing our method distributively.

The second point of comparison is approximation accuracy. The accuracy of standard finite precision matrix inversion is controlled by the number of bits of precision used by the multiplier. On the other hand, by design, our proposed algorithm introduces an additional approximation error due to its reliance on successive approximation iterations. As we showed numerically though in Figure 1, after a few iterations of Algorithm 1 with CG; we can achieve the same error rate as when exact matrix inversion is used, with a lower complexity. Furthermore, in building risk minimizing ML models, approximate solutions using iterative approximations are often faster and sufficient for achieving desired performance benchmarks. Additionally, the approximation accuracy of our proposed matrix inversion method is controllable by adjusting the number of iterations carried out locally by the workers.

Lastly, we discuss when Algorithm 1 might have advantages over exact matrix inversion in terms of computational complexity and waiting time. For simplicity, we assume that exact computation of $\mathbf{A}^{-1}$ requires $\mathcal{O}(N^{2.372})$ operations. When utilizing our algorithm with CG on a single server, we require $\mathcal{O}(\tilde{N}N^2)$ operations to guarantee convergence. Thus, in this case; Algorithm 1 is beneficial when $\tilde{N} < N^{0.372}$, where $\tilde{N}$ is the number of distinct eigenvalues of $\mathbf{A}^\top\mathbf{A}$. When employing a distributed implementation, in terms of the waiting time through phase (c); our approach is beneficial when $\tilde{N} < N^{0.872}$.

VI. Conclusion and Future Work

In this paper, we addressed the problem of approximate computation of the inverse of a matrix distributively in a relaxed FL setting, under the possible presence of straggling workers. We provided approximation error bounds for our approach, as well as security and recovery guarantees. We also provided numerical experiments that validated our proposed approach.

There are several interesting future directions. One avenue to consider is incorporating fully homomorphic encryption in our phases (b),(c),(d), to obtain a FL scheme; and prevent the requirement of clients need to recover each others' information. An important issue is the numerical stability of the BRS approach, so exploring other suitable generator matrices could be beneficial; *e.g.* circulant permutation and rotation matrices [46]. It is also worth investigating if we can reduce the communication rounds when computing the pseudoinverse through our approach. This depends on the CMM which is being utilized, though using different ones for each of the two multiplications may also be beneficial.

In terms of coding-theory, it would be interesting to see if it is possible to reduce the complexity of our decoding step. Specifically, could well-known RS decoding algorithms such as the Berlekamp-Welch algorithm be exploited? Another direction, is leveraging approximate CCMs. The work of [47] considers the GC problem for *approximate* and *exact* recovery through Lagrange interpolation, for heterogeneous workers in the presence of stragglers and adversaries. A potential scheme for matrix inversion could also be developed through the methods of [47]. In terms of our approximation algorithms, an avenue worth exploring is that of incorporating approximate and/or sparse Gaussian elimination [48], [49] into our distributed CCM.

Tribute to Alex Vardy: As this is a special issue dedicated to the memory Alexander Vardy, we mention how this paper relates to his work. Even though Alex had not worked on CC, his contributions to RS codes are immense. A focus of ours is to reduce the decoding complexity of the proposed BRS-based CCM, while in [50] it was shown by Guruswami and Vardy that maximum-likelihood decoding of RS codes is NP-hard. Another highly innovative work of Vardy's is [51], in which the 'Parvaresh-Vardy codes' were introduced; and the associated list-decoding algorithm was shown to yield an improvement over the Guruswami–Sudan algorithm. This was subsequently improved by Guruswami and Rudra [52], whose techniques were exploited in [26] to introduce list-decoding in CC.

APPENDIX A
ADDITIONAL MATERIAL AND BACKGROUND

In this appendix, we include material and background which was used in our derivations. First, we recall what an ϵ-*optimal solution/point* is, which was used in the proof of Proposition 4. Next, we state the MDS theorem and the BCH Bound. We then give a brief overview of the GC scheme from [10], to show how it differs from our CMIM. We also explicitly give their construction of a balanced mask matrix $\mathbf{M} \in \{0,1\}^{n \times k}$, which we use for the construction of the BRS generator matrices. Lastly, we illustrate a simple example of the encoding matrix.

Definition 9 ([53]). *A point $\bar{x}$ is said to be an ϵ-**optimal solution/point** to a minimization problem with objective function $f(x)$, if for any x, it holds that $f(x) \geqslant f(\bar{x}) - \epsilon$, where $\epsilon \geqslant 0$. When $\epsilon = 0$, an ϵ-optimal solution is an exact minimizer.*

Theorem 10 (MDS Theorem — [44]). *Let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$, with $\mathbf{G}, \mathbf{K}$ the generator and parity-check matrices. Then, the following are equivalent:*

1) *$\mathcal{C}$ is a MDS code, i.e. $d = n - k + 1$*
2) *every set of $n - k$ columns of $\mathbf{K}$ is linearly independent*
3) *every set of k columns of $\mathbf{G}$ is linearly independent*
4) *$\mathcal{C}^{\perp}$ is a MDS code.*

Theorem 11 (BCH Bound — [17], [33]). *Let $p(\mathsf{x}) \in \mathbb{F}_q[\mathsf{x}] \backslash \{0\}$ with t cyclically consecutive roots, i.e. $p(\alpha^{j+\iota}) = 0$ for all $\iota \in \mathbb{N}_t$. Then, at least $t + 1$ coefficients of $p(\mathsf{x})$ are nonzero.*

Algorithm 2: $\mathrm{MaskMatrix}(n, k, d)$ [10]

Input: $n, k, d \in \mathbb{Z}_+$ s.t. $n > d, k$ and $w = \frac{kd}{n}$
Output: row-balanced mask matrix $\mathbf{M} \in \{0,1\}^{n \times k}$
$\mathbf{M} \leftarrow \mathbf{0}_{n \times k}$
for $j = 0$ *to* $k - 1$ **do**
    for $i = 0$ *to* $d - 1$ **do**
        $\iota \leftarrow (i + jd + 1) \bmod n$
        $\mathbf{M}_{r,\iota} \leftarrow 1$
    end
end
return $\mathbf{M}$

A. Generator Matrix Example

For an example, consider the case where $n = 9$, $k = 6$ and $d = 6$, thus $w = \frac{kd}{n} = 4$. Then, Algorithm 2 produces

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \in \{0,1\}^{9 \times 6}.$$

For our CCM, this means that the ith worker computes the blocks indexed by $\mathrm{supp}(\mathbf{M}_{(i)})$, *e.g.* $\mathrm{supp}(\mathbf{M}_{(1)}) = \{1, 2, 4, 5\}$. We denote the indices of the respective task allocations by $\mathcal{J}_i = \mathrm{supp}(\mathbf{M}_{(i)})$. The entries of the generator matrix $\mathbf{G}$ are the evaluations of the constructed polynomials (3) at each of the evaluation points $\mathcal{B} = \{\beta_i\}_{i=1}^n$, *i.e.* $\mathbf{G}_{ij} = p_j(\beta_i)$. This results in:

$$\mathbf{G} = \begin{pmatrix} p_1(\beta_1) & p_2(\beta_1) & 0 & p_4(\beta_1) & p_5(\beta_1) & 0 \\ p_1(\beta_2) & p_2(\beta_2) & 0 & p_4(\beta_2) & p_5(\beta_2) & 0 \\ p_1(\beta_3) & p_2(\beta_3) & 0 & p_4(\beta_3) & p_5(\beta_3) & 0 \\ p_1(\beta_4) & 0 & p_3(\beta_4) & p_4(\beta_4) & 0 & p_6(\beta_4) \\ p_1(\beta_5) & 0 & p_3(\beta_5) & p_4(\beta_5) & 0 & p_6(\beta_5) \\ p_1(\beta_6) & 0 & p_3(\beta_6) & p_4(\beta_6) & 0 & p_6(\beta_6) \\ 0 & p_2(\beta_7) & p_3(\beta_7) & 0 & p_5(\beta_7) & p_6(\beta_7) \\ 0 & p_2(\beta_8) & p_3(\beta_8) & 0 & p_5(\beta_8) & p_6(\beta_8) \\ 0 & p_2(\beta_9) & p_3(\beta_9) & 0 & p_5(\beta_9) & p_6(\beta_9) \end{pmatrix} .$$

APPENDIX B
DISTRIBUTED PSEUDOINVERSE

For full-rank rectangular matrices $\mathbf{A} \in \mathbb{R}^{N \times M}$ where $N > M$, one resorts to the left Moore–Penrose pseudoinverse $\mathbf{A}^\dagger \in \mathbb{R}^{M \times N}$, for which $\mathbf{A}^\dagger \mathbf{A} = \mathbf{I}_M$. In Algorithm 3, we present how to approximate the left pseudoinverse of $\mathbf{A}$, by using the fact that $\mathbf{A}^\dagger = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top$; since $\mathbf{A}^\top \mathbf{A} \in \mathrm{GL}_M(\mathbb{R})$. The right pseudoinverse $\mathbf{A}^\dagger = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1}$ of $\mathbf{A} \in \mathbb{R}^{M \times N}$ where $M < N$, can be obtained by a modification of Algorithm 3.

Just like the inverse, the pseudoinverse of a matrix also appears in a variety of applications. Computing the pseudoinverse of $\mathbf{A} \in \mathbb{R}^{N \times M}$ for $N > M$ is even more cumbersome, as it requires inverting the Gram matrix $\mathbf{A}^\top \mathbf{A}$. For this appendix, we consider a full-rank matrix $\mathbf{A}$.

One could naively attempt to modify Algorithm 1 in order to retrieve $\mathbf{A}^\dagger$ such that $\mathbf{A}^\dagger \mathbf{A} = \mathbf{I}_M$, by approximating the rows of $\mathbf{A}^\dagger$. This would *not* work, as the underlying optimization problems would not be strictly convex. Instead, we use Algorithm 3 to estimate the rows of $\mathbf{B}^{-1} := (\mathbf{A}^\top \mathbf{A})^{-1}$, and then multiply the estimate $\widehat{\mathbf{B}^{-1}}$ by $\mathbf{A}^\top$. This gives us the approximation $\widehat{\mathbf{A}^\dagger} = \widehat{\mathbf{B}^{-1}} \cdot \mathbf{A}^\top$.

The drawback of Algorithm 3 is that it requires two additional matrix multiplications, $\mathbf{A}^\top \mathbf{A}$ and $\widehat{\mathbf{B}^{-1}} \mathbf{A}^\top$. We overcome this barrier by using a CMM scheme twice, to recover $\widehat{\mathbf{A}^\dagger}$ in a two or three-round communication CC approach. These are discussed in below.

Bounds on $\mathrm{err}_F(\widehat{\mathbf{A}^{-1}})$ and $\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^{-1}})$ can be established for both algorithms, specific to the black-box least squares solver being utilized.

Algorithm 3: Estimating $\mathbf{A}^\dagger$

Input: full-rank $\mathbf{A} \in \mathbb{R}^{N \times M}$ where $N > M$
$\mathbf{B} \leftarrow \mathbf{A}^\top \mathbf{A}$
for *i=1 to M* **do**
$\quad \hat{\mathbf{c}}_i = \arg\min_{\mathbf{c} \in \mathbb{R}^{1 \times M}} \left\{ g_i(\mathbf{c}) := \|\mathbf{c}\mathbf{B} - \mathbf{e}_i^\top\|_2^2 \right\}$
$\quad \hat{\mathbf{b}}_i \leftarrow \hat{\mathbf{c}}_i \cdot \mathbf{A}^\top$
end
return $\widehat{\mathbf{A}^\dagger} \leftarrow \left[\hat{\mathbf{b}}_1^\top \quad \cdots \quad \hat{\mathbf{b}}_M^\top \right]^\top$ $\qquad\qquad\qquad\qquad\qquad \triangleright \widehat{\mathbf{A}^\dagger}_{(i)} = \hat{\mathbf{b}}_i$

Corollary 12. *For full-rank $\mathbf{A} \in \mathbb{R}^{N \times M}$ with $N > M$, we have $\mathrm{err}_F(\widehat{\mathbf{A}^\dagger}) \leqslant \frac{\sqrt{M}\epsilon \cdot \kappa_2}{\sqrt{2}\sigma_{\min}(\mathbf{A})^3}$ and $\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^\dagger}) \leqslant \frac{\sqrt{M}\epsilon \cdot \kappa_2}{\sqrt{2}\sigma_{\min}(\mathbf{A})^2}$ when using SD to solve the subroutine optimization problems of Algorithm 3, with termination criteria $\|\nabla g_i(\mathbf{c}^{[t]})\|_2 \leqslant \epsilon$.*

Proof. From (10), it follows that

$$\|\mathbf{B}^{-1}\mathbf{e}_i - \hat{\mathbf{c}}_i^\top\|_2 \leqslant \frac{\epsilon/\sqrt{2}}{\sigma_{\min}(\mathbf{B})^2} = \frac{\epsilon/\sqrt{2}}{\sigma_{\min}(\mathbf{A})^4} =: \delta .$$

The above bound implies that for each summand of the Frobenius error; $\|\hat{\mathbf{b}}_i - \mathbf{A}_{(i)}^\dagger\|_2 = \|\hat{\mathbf{c}}_i \mathbf{A}^\top - \mathbf{e}_i^\top \cdot \mathbf{B}^{-1} \mathbf{A}^\top\|_2$, we have $\|\hat{\mathbf{b}}_i - \mathbf{A}_{(i)}^\dagger\|_2 \leqslant \delta \|\mathbf{A}^\top\|_2$. Summing the right hand side M times, we get that

$$\mathrm{err}_F(\widehat{\mathbf{A}^\dagger})^2 \leqslant M \cdot (\delta \|\mathbf{A}^\top\|_2)^2$$
$$= \frac{M\epsilon^2 \cdot \sigma_{\max}(\mathbf{A})^2}{2\sigma_{\min}(\mathbf{A})^8}$$

$$= \frac{M\epsilon^2 \cdot \kappa_2^2}{2\sigma_{\min}(\mathbf{A})^6} \ .$$

By taking the square root, we have shown the first claim.

Since $1/\sigma_{\min}(\mathbf{A}) = \|\mathbf{A}^\dagger\|_2 \leqslant \|\mathbf{A}^\dagger\|_F$, it then follows that

$$\mathrm{err}_{\mathrm{r}F}(\widehat{\mathbf{A}^\dagger}) = \frac{\mathrm{err}_F(\widehat{\mathbf{A}^\dagger})}{\|\mathbf{A}^\dagger\|_F} \leqslant \frac{\mathrm{err}_F(\widehat{\mathbf{A}^\dagger})}{\|\mathbf{A}^\dagger\|_2} = \frac{\sqrt{M}\epsilon \cdot \kappa_2}{\sqrt{2}\sigma_{\min}(\mathbf{A})^2} \ ,$$

which completes the proof. ∎

A. Pseudoinverse from Polynomial CMM

One approach to leverage Algorithm 3 in a two-round communication scheme is to first compute $\mathbf{B} = \mathbf{A}^\top \mathbf{A}$ through a CMM scheme, then share $\mathbf{B}$ with all the workers who estimate the rows of $\widehat{\mathbf{B}^{-1}}$, and finally use another CMM to locally encode the estimated columns with blocks of $\mathbf{A}^\top$; to recover $\widehat{\mathbf{A}^\dagger} = \widehat{\mathbf{B}^{-1}} \cdot \mathbf{A}^\top$. Even though there are only two rounds of communication, the fact that we have a local encoding by the workers results in a higher communication load overall. An alternative approach which circumvents this issue, uses three-rounds of communication.

For this approach, we use the polynomial CMM scheme from [8] twice, along with our coded matrix inversion scheme. This CMM has a reduced communication load, and minimal computation is required by the workers. To have a consistent recovery threshold across our communication rounds, we partition $\mathbf{A}$ as in (11) into $\bar{k} = \sqrt{n-s} = \sqrt{k}$ blocks. Each block is of size $N \times \bar{T}$, for $\bar{T} = \frac{M}{k}$. The encodings from [8] of the partitions $\{\mathbf{A}_\iota\}_{\iota=1}^{\bar{k}}$ for carefully selected parameters $a, b \in \mathbb{Z}_+$ and distinct elements $\gamma_i \in \mathbb{F}_q$, are

$$\tilde{\mathbf{A}}_i^a = \sum_{j=1}^k \mathbf{A}_j \gamma_i^{(j-1)a} \quad \text{and} \quad \tilde{\mathbf{A}}_i^b = \sum_{j=1}^k \mathbf{A}_j \gamma_i^{(j-1)b}$$

for each worker indexed by i. Thus, each encoding is comprised of $N\bar{T}$ symbols. The workers compute the product of their respective encodings $(\tilde{\mathbf{A}}_i^a)^\top \cdot \tilde{\mathbf{A}}_i^b$. The decoding step corresponds to an interpolation step, which is achievable when $\bar{k}^2 = k$ many workers respond[5], which is the optimal recovery threshold for CMM. Any fast polynomial interpolation or RS decoding algorithm can be used for this step, to recover $\mathbf{B}$.

Next, the master shares $\mathbf{B}$ with all the workers (from V-A, this is necessary), who are requested to estimate the *column-blocks* of $\widehat{\mathbf{B}^{-1}}$

$$\widehat{\mathbf{B}^{-1}} = \begin{bmatrix} \bar{\mathcal{B}}_1 & \cdots & \bar{\mathcal{B}}_k \end{bmatrix} \quad \text{where } \bar{\mathcal{B}}_j \in \mathbb{R}^{M \times \bar{T}} \ \forall j \in \mathbb{N}_k \tag{15}$$

according to Algorithm 1. We can then recover $\widehat{\mathbf{B}^{-1}}$ by our BRS based scheme, once k workers send their encoding.

For the final round, we encode $\widehat{\mathbf{B}^{-1}}$ as

$$\tilde{\mathbf{B}}_i^a = \sum_{j=1}^k \bar{\mathcal{B}}_j \gamma_i^{(j-1)a}$$

which are sent to the respective workers. The workers already have in their possession the encodings $\tilde{\mathbf{A}}_i^b$. We then carry out the polynomial CMM where each worker is requested to send back $(\tilde{\mathbf{B}}_i^a)^\top \cdot \tilde{\mathbf{A}}_i^b$. The master server can then recover $\widehat{\mathbf{A}^\dagger}$.

Theorem 13. *Consider $\mathbf{G} \in \mathbb{F}^{n \times k}$ as in Theorem 7. By using any CMM, we can devise a matrix pseudoinverse CCM by utilizing Algorithm 3, in two-rounds of communication. By using polynomial CMM [8], we achieve this with a reduced communication load and minimal computation, in three-rounds of communication.*

REFERENCES

[1] B. G. Greenberg and A. E. Sarhan, "Matrix inversion, its interest and application in analysis of data," *Journal of the American Statistical Association*, vol. 54, no. 288, pp. 755–766, 1959.

[2] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. USA: Society for Industrial and Applied Mathematics, 2002.

[3] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[4] 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 *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1215–1225.

[5] C. Karakus, Y. Sun, and S. Diggavi, "Encoded Distributed Optimization," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2890–2894.

[6] N. Charalambides, H. Mahdavifar, M. Pilanci, and A. O. Hero, "Orthonormal Sketches for Secure Coded Regression," in *2022 IEEE International Symposium on Information Theory (ISIT)*, 2022, pp. 826–831.

[7] S. Li and S. Avestimehr, "Coded Computing," *Foundations and Trends® in Communications and Information Theory*, vol. 17, no. 1, 2020.

[8] 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*, 2017, pp. 4403–4413.

---

[5]We select $\bar{k} = \sqrt{k}$ in the partitioning of $\mathbf{A}$ in (11) when deploying this CMM, to attain the same recovery threshold as our inversion scheme.

[9] N. Charalambides, M. Pilanci, and A. O. Hero, "Secure Linear MDS Coded Matrix Inversion," in *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2022, pp. 1–8.

[10] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving Distributed Gradient Descent Using Reed-Solomon Codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 2027–2031.

[11] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded Federated Learning," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[12] S. Prakash, S. Dhakal, M. R. Akdeniz, Y. Yona, S. Talwar, S. Avestimehr, and N. Himayat, "Coded Computing for Low-Latency Federated Learning over Wireless Edge Networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 233–250, 2020.

[13] R. Schlegel, S. Kumar, E. Rosnes, and A. G. i. Amat, "CodedPaddedFL and CodedSecAgg: Straggler Mitigation and Secure Aggregation in Federated Learning," *arXiv e-prints*, pp. arXiv–2112, 2021.

[14] S. Kumar, R. Schlegel, E. Rosnes, and A. G. i. Amat, "Coding for Straggler Mitigation in Federated Learning," *arXiv preprint arXiv:2109.15226*, 2021.

[15] M. Xhemrishi, A. G. i. Amat, E. Rosnes, and A. Wachter-Zeh, "Computational Code-Based Privacy in Coded Federated Learning," *arXiv preprint arXiv:2202.13798*, 2022.

[16] S. Ha, J. Zhang, O. Simeone, and J. Kang, "Coded Federated Computing in Wireless Networks with Straggling Devices and Imperfect CSI," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 2649–2653.

[17] W. Halbawi, Z. Liu, and B. Hassibi, "Balanced Reed-Solomon Codes," in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 935–939.

[18] ——, "Balanced Reed-Solomon Codes for all parameters," in *2016 IEEE Information Theory Workshop (ITW)*. IEEE, 2016, pp. 409–413.

[19] N. Charalambides, H. Mahdavifar, and A. O. Hero, "Numerically Stable Binary Gradient Coding," *arXiv preprint arXiv:2001.11449*, 2020.

[20] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated Optimization: Distributed Machine Learning for On-Device Intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[21] Y. Yang, P. Grover, and S. Kar, "Coded Distributed Computing for Inverse Problems," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017, pp. 709–719.

[22] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning*, 2017, pp. 3368–3376.

[23] A. M. Subramaniam, A. Heidarzadeh, A. K. Pradhan, and K. R. Narayanan, "Product Lagrange Coded Computing," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 197–202.

[24] M. Fahim and V. R. Cadambe, "Lagrange Coded Computing with Sparsity Constraints," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2019, pp. 284–289.

[25] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog Lagrange Coded Computing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 283–295, 2021.

[26] M. Soleymani, R. E. Ali, H. Mahdavifar, and A. S. Avestimehr, "List-Decodable Coded Computing: Breaking the Adversarial Toleration Barrier," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 867–878, 2021.

[27] J. Zhu and S. Li, "Generalized Lagrange Coded Computing: A Flexible Computation-Communication Tradeoff," in *2022 IEEE International Symposium on Information Theory (ISIT)*, 2022, pp. 832–837.

[28] S. Kiani and S. C. Draper, "Successive Approximation Coding for Distributed Matrix Multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 2, pp. 286–305, 2022.

[29] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the Optimal Recovery Threshold of Coded Matrix Multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2019.

[30] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[31] S. H. Dau, W. Song, Z. Dong, and C. Yuen, "Balanced Sparsest Generator Matrices for MDS Codes," in *2013 IEEE International Symposium on Information Theory*, 2013, pp. 1889–1893.

[32] M. Krause, "A Simple Proof of the Gale-Ryser Theorem," *The American Mathematical Monthly*, vol. 103, no. 4, pp. 335–337, 1996.

[33] R. J. McEliece, *Theory of Information and Coding*, 2nd ed. USA: Cambridge University Press, 2001.

[34] V. V. Williams, Y. Xu, Z. Xu, and R. Zhou, "New Bounds for Matrix Multiplication: from Alpha to Omega," *arXiv preprint arXiv:2307.07970*, 2023.

[35] Å. Björck and V. Pereyra, "Solution of Vandermonde Systems of Equations," *Mathematics of Computation*, vol. 24, pp. 893–903, 1970.

[36] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[37] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," *Carnegie Mellon University, Tech. Rep.*, 1994.

[38] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[39] S. Bubeck, "Convex Optimization: Algorithms and Complexity," *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015. [Online]. Available: http://dx.doi.org/10.1561/2200000050

[40] R. M. Gower, "Sketch and Project: Randomized Iterative Methods for Linear Systems and Inverting Matrices," *arXiv preprint arXiv:1612.06013*, 2016.

[41] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006.

[42] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[43] G. R. Blakley, "Safeguarding cryptographic keys," *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pp. 313–318, 1899.

[44] S. Ling and C. Xing, *Coding Theory: A First Course*. Cambridge University Press, 2004.

[45] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient Coding from Cyclic MDS Codes and Expander Graphs," *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7475–7489, 2020.

[46] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," *IEEE Transactions on Information Theory*, vol. 68, no. 4, pp. 2684–2703, 2021.

[47] T. Jahani-Nezhad and M. A. Maddah-Ali, "Optimal Communication-Computation Trade-Off in Heterogeneous Gradient Coding," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 1002–1011, 2021.

[48] R. Kyng and S. Sachdeva, "Approximate Gaussian Elimination for Laplacians — Fast, Sparse, and Simple," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 573–582.

[49] R. Kyng, "Approximate Gaussian Elimination," Ph.D. dissertation, PhD thesis. Yale University, 2017.

[50] V. Guruswami and A. Vardy, "Maximum-Likelihood Decoding of Reed-Solomon Codes is NP-hard," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2249–2256, 2005.

[51] F. Parvaresh and A. Vardy, "Correcting Errors Beyond the Guruswami-Sudan Radius in Polynomial Time," in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. IEEE, 2005, pp. 285–294.

[52] V. Guruswami and A. Rudra, "Explicit Codes Achieving List Decoding Capacity: Error-Correction With Optimal Redundancy," *IEEE Transactions on Information Theory*, vol. 54, no. 1, pp. 135–150, 2008.

[53] F. Bai, Z. Wu, and D. Zhu, "Sequential Lagrange multiplier condition for ϵ-optimal solution in convex programming," *Optimization*, vol. 57, no. 5, pp. 669–680, 2008.