Byzantine-Resilient Federated PCA and Low-Rank Column-Wise Sensing

Ankit Pratap Singh and Namrata Vaswani, *Fellow, IEEE*

***Abstract*— This work considers two related learning problems in a federated attack-prone setting – federated principal components analysis (PCA) and federated low rank column-wise sensing (LRCS). The node attacks are assumed to be Byzantine which means that the attackers are omniscient and can collude. We introduce a novel provably Byzantine-resilient communication-efficient and sample-efficient algorithm, called Subspace-Median, that solves the PCA problem and is a key part of the solution for the LRCS problem. We also study the most natural Byzantine-resilient solution for federated PCA, a geometric median based modification of the federated power method, and explain why it is not useful. Our second main contribution is a complete alternating gradient descent (GD) and minimization (altGDmin) algorithm for Byzantine-resilient horizontally federated LRCS and sample and communication complexity guarantees for it. Extensive simulation experiments are used to corroborate our theoretical guarantees. The ideas that we develop for LRCS are easily extendable to other LR recovery problems as well.**

***Index Terms*— Federated PCA, Byzantine, matrix sensing, linear representation learning.**

I. Introduction

FEDERATED learning is a setting where multiple entities/nodes/clients collaborate in solving a machine learning (ML) problem. Each node can only communicate with a central server or service provider that we refer to as "center" in this paper. The data observed or measured at each node/client is stored locally and should not be shared with the center. Summaries of it can be shared with the center. The center typically aggregates the received summaries and broadcasts the aggregate to all the nodes [3]. One of the challenges in this setup is adversarial attacks on the nodes. In this work we assume Byzantine attacks, i.e., the adversarial nodes are omniscient and can collude [4], [5], [6], [7], [8]. "Omniscient" means that the attacking nodes have knowledge of all the data at every node and the exact algorithm (and all its parameters) implemented by every node, including center, and can use this information to design the worst possible attacks at each algorithm iteration.

This work develops provably Byzantine resilient algorithms for solving two related problems – federated principal components analysis (PCA) and horizontally-federated low rank (LR) column-wise sensing or LRCS – in a communication- and sample-efficient fashion. The first goal in solving both problems is to reliably estimate the subspace spanned by the top r singular vectors of an unknown symmetric $n \times n$ matrix, $\mathbf{\Phi}^*$. In case of PCA, $\mathbf{\Phi}^*$ is the population covariance matrix of the available data. For each $\ell = 1, 2, \ldots, L$, node ℓ observes an $n \times q_\ell$ data matrix $\boldsymbol{D}_\ell$ which can be used to compute an estimate $\mathbf{\Phi}_\ell := \boldsymbol{D}_\ell \boldsymbol{D}_\ell^\top / q_\ell$ of $\mathbf{\Phi}^*$. PCA is well known to have a large number of applications in scientific visualization and as a pre-processing step for speeding up various ML tasks. LRCS finds applications in accelerated dynamic MRI [9], [10], multi-task linear representation learning and few shot learning [2], [11], [12], and federated sketching [13], [14], [15].

A. Existing Work

1) Byzantine-Resilient Federated Machine Learning (ML): There has been a large amount of recent work on Byzantine-resilient federated ML algorithms, some of which come with provable guarantees [6], [7], [8], [16], [17], [17], [18], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29]. Some of the theoretical guarantees are asymptotic, and almost all of them analyze the standard gradient descent (GD) algorithm or stochastic GD. Typical solutions involve replacing the mean/sum of the gradients from the different nodes by a different robust statistic, such as geometric median (of means) [16], trimmed mean, coordinate-wise mean [8] or Krum [6].

One of the first non-asymptotic results for Byzantine attacks is [16]. This used the geometric median (GM) of means to replace the regular mean/sum of the partial gradients from each node. Under standard assumptions (strong convexity, Lipschitz gradients, sub-exponential-ity of sample gradients, and an upper bound on the fraction of Byzantine nodes), it provided an exponentially decaying bound on the distance between the estimate at the t-th iteration and the unique global minimizer. In follow-up work [8], the authors studied the coordinate-wise mean and the trimmed-mean estimators and developed guarantees for both convex and non-convex problems. Because these works used coordinate-wise estimators, their results needed smoothness and convexity along each dimension. This is

Manuscript received 22 September 2023; revised 25 May 2024; accepted 29 July 2024. Date of publication 21 August 2024; date of current version 22 October 2024. This work was supported by the NSF under Grant CIF-2115200. An earlier version of this paper was presented in part at ISIT 2024 [DOI: 10.1109/ISIT57864.2024.10619161] and in part at ICML 2024 [2]. *(Corresponding author: Namrata Vaswani.)*

The authors are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: sankit@iastate.edu; namrata@iastate.edu).

Communicated by Y. Chi, Associate Editor for Machine Learning and Statistics.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2024.3442211.

Digital Object Identifier 10.1109/TIT.2024.3442211

a stronger, and sometimes impractical, assumption. Another interesting series of works [7], [19] provides non-asymptotic guarantees for Byzantine resilient stochastic GD. This work develops an elaborate median based algorithm to detect and filter out the Byzantine nodes. The theoretical analysis assumes that the norm of the sample gradients is bounded by a constant that does not depend on the gradient dimension. This can be a restrictive assumption. With this assumption, they are able to obtain sample complexity guarantees that do not depend on the signal dimension. These works also assume that the set of Byzantine nodes is the same for all GD iterations, while the work of [16] allowed this set to vary at each iteration.

Most of the above works considered the homogeneous data setting; this means that the data that is i.i.d. (independent and identically distributed) across all nodes. More recent work has focused on heterogeneous distributions (data is independent across nodes but is not identically distributed) and proved results under upper bounds on the amount of heterogeneity [23], [24], [25], [26]. Other works rely on detection methods to handle heterogeneous gradients [17], [18], [27], [28], [29]. These assume the existence of a trustworthy root/validation dataset at the central server that is used for detecting the adversarial gradients.

2) Work on Robust PCA and Subspace Learning and Tracking, and Other Robust Estimation Problems: There is much work also on solving the robust PCA problem using the low rank plus sparse (L+S) [30], [31], [32], [33], [34] or other models, on robust subspace learning [35], [36], and on robust subspace tracking problems [37], [38], [39]. The review article [40] provides a comprehensive summary of the older work. In addition there is other related work that uses the median or vector medians for other types of outlier robust algorithms, e.g., [41]. However, there are two key differences between all these works and the problem that we study in this paper. (1) All of these works assume that the outlier or the attack is on the observed or measured data. In security literature, such attacks (in which only the data can be corrupted) are referred to as "data-poisoning" attacks. The algorithms from these works cannot be used to deal with Byzantine attacks which involve corruption of the (intermediate and/or final) algorithm estimates sent by some nodes. (2) Secondly, almost all of these are designed for the centralized setting. A possible way to extend any of these ideas to the federated setting is for the nodes to share their raw data with the center and for the center to implement the same algorithm as that developed in these works. However, this would not be communication-efficient[1]. To distinguish from the L+S, or other, model-based robust PCA work, here, we use the term "resilient" to denote attack-resilience.

3) Work on Robust Statistics - Robust Mean and Robust Covariance Estimation: There is a large amount of existing work in the general robust statistics literature, most of it is on robust mean estimation, and some on robust covariance estimation, e.g., see [42], [43], [44], [45], and [46]. None of these can be extended to solving our problem in a communication-efficient fashion and most of these also have much larger sample complexities. As an example, the work of Minsker [46] studies the geometric median (GM), which is one well-known approach to compute a reliable estimate of a vector-valued quantity using multiple individual estimates of it when some of these estimates may be corrupted by outliers [16], [46]. In [46, Corollary 4.3], Minsker shows the application of GM for "robust PCA" - provably accurate robust/resilient covariance estimation followed by SVD on the robust covariance estimate to compute its top r singular vectors. We refer to this solution as *SVD-ResCovEst*. This approach needs order $q_\ell \geq n^2$ samples. Moreover, it cannot be federated efficiently because it requires that each node ℓ shares $\boldsymbol{D}_\ell \boldsymbol{D}_\ell^\top$ with the center. This has a communication cost of order n^2. A similar discussion applies for the result of [42, Theorem 4.35] as well.

4) Work on the LR Column-Wise Sensing (LRCS) Problem: The LRCS problem, and its phaseless measurements' extension, LR phase retrieval, have been extensively studied in recent years [10], [15], [47], [48], [49], [50], mostly in centralized settings. The work of [49] and [50] introduced a fast and communication-efficient solution to attack-free federated LRCS, called alternating GD and minimization (altGDmin). AltGDmin is initialized using spectral initialization.

5) Federated PCA and Subspace Learning; No Attacks: There is also somewhat related work on federated PCA and subspace learning that does not consider any attacks or other outliers, e.g., [51], [52], and [53].

B. Our Contributions

A natural way to make the SVD-ResCovEst approach communication-efficient is to borrow ideas from the sketching literature and share $\boldsymbol{\Phi}_\ell \boldsymbol{U}_{any}$ for some (possibly random) $n \times r$ matrix $\boldsymbol{U}_{any}$. This idea is, in fact, one iteration of the power method for computing the r-SVD of a matrix [54], [55]. It can be converted into a provably correct solution by using the GM to modify the power method. We refer to this solution as Resilient Power Method (ResPowMeth), and obtain a set of sufficient conditions for it to work. We show that this approach is both provably resilient to Byzantine attacks and communication efficient under certain restrictive assumptions on the accuracy of the individual nodes' partial covariance estimates, which translate into a very large sample complexity for PCA: for n-length data vectors, ResPowMeth works if $q_\ell \geq Cn^2r^2$.

Our first important contribution is a novel and well-motivated solution to Byzantine-resilient federated subspace estimation, and PCA, that is both communication-efficient and sample-efficient. We refer to this as "Subspace-Median". Its guarantee is provided in Theorem 3.1 and Corollary 1. We show how the Subspace Median can be used to provably solve two practically useful problems: (i) Byzantine resilient federated PCA, and (ii) the initialization step of Byzantine-resilient horizontal federated LRCS. For the PCA problem, we show that this works well

[1]One exception is the work of [37] that considers the federated setting. However this has two important limitations: (i) it assumes that, for the initialization step, the data is outlier-free; (ii) and, it requires a much larger number of observed samples than what traditional LR matrix completion literature needs.

with just $q_\ell \geq Cnr$ samples. We also develop Subspace Median-of-Means (MoM) extensions for both problems. These help improve the sample complexity at the cost of reduced Byzantine/outlier tolerance. For all these algorithms, Theorem 3.1 helps prove sample, communication, and time complexity bounds for ϵ-accurate subspace recovery. Extensive simulation experiments corroborate our theoretical results.

Our second important contribution is a provable communication-efficient and sample-efficient complete solution to horizontally federated LRCS. For solving it, we develop a GM-based modification of the alternating GD and minimization (altGDmin) algorithm from earlier work [49]. We use Subspace Median and Subspace MoM to make its spectral initialization step Byzantine-resilient. For the complete algorithm, we can show that it obtains an ϵ-accurate estimate of the unknown LR matrix using only order $nr^2 \log(1/\epsilon)$ samples per node, and with a total communication cost of only order $nr \log(1/\epsilon)$ per node. Both costs are comparable to what basic altGDmin needs for solving this problem in the attack-free setting [49], [50].

The overall approach that we develop for modifying the altGDmin algorithm (use Subspace-Median for initialization and GM of gradients for the GD step), and analyzing it, can be extended to make GD-based solutions to many other similar non-convex problems in federated settings Byzantine-resilient. Some examples include vertically federated LRCS, LR matrix completion, LR phase retrieval, LR plus sparse matrix recovery (robust PCA). Our approach for analyzing Byzantine-resilient PCA is also extendable to solving PCA for approximately LR datasets, PCA for such datasets with missing entries (see Remark 4), and also to subspace tracking and robust subspace tracking. We describe these in Sec. VIII-A.

C. Novelty of Our Algorithmic and Proof Techniques

While both SVD and geometric median (GM) are well known in literature, we are not aware of any notions of "median" for subspaces. We cannot directly use the GM on the subspace basis matrices because these do not lie in a Euclidean space, e.g., $\boldsymbol{U}$, $-\boldsymbol{U}$ specify the same subspace even though $\|\boldsymbol{U} - (-\boldsymbol{U})\|_F = 2\sqrt{r} \neq 0$. The design of Subspace Median relies on the fact that the Frobenius norm of the difference between two subspace projection matrices is within a constant factor of the subspace distance between their respective subspaces. Its analysis also uses the fact that these projection matrices are bounded by $\sqrt{r}$ in Frobenius norm. We use these facts and Lemma 4 (GM lemma for bounded inputs) to prove our key lemma, Lemma 1. This is combined with the Davis-Kahan $\sin\Theta$ theorem to prove Theorem 3.1. This result is likely to be widely applicable in making various other subspace recovery problems Byzantine resilient.

Our analysis of the AltGDmin iterations relies heavily on the lemmas proved in [49] and the overall simplified proof approach developed in [50]. However, we need to modify this approach to deal with the fact that we compute the geometric median of the gradients from the different nodes. The GM analysis provides bounds on Frobenius norms, and hence our analysis also uses the Frobenius norm subspace distance instead of the 2-norm one; see Lemma 8. At the same time it avoids the complicated proof approach (does not need to use the fundamental theorem of calculus) of [49]. The main new step is the bound on the difference between the expected values of the gradients from two good nodes conditioned on past estimates and data[2]. See Lemma 2. This lemma is used along with Lemma 6 (our GM lemma for potentially unbounded inputs) to obtain Lemma 10. This discussion will be clearer from the proof outline provided below Theorem 5.3.

D. Organization

We define the problems, the notation, and introduce the geometric median in the next section. Sec. III develops the Subspace Median and Resilient Power Method (ResPowMeth) solutions, and provides their theoretical guarantees. Sec. IV develops corollaries for the resilient PCA problem, compares the three approaches – SVD-ResCovEst [46], ResPowMeth, and Subspace Median. A summary is provided in Table I. Subspace Median of Means is also developed here. Sec. V develops a complete altGDmin-based solution for resilient horizontally federated LRCS. Proofs for Sections III and IV are provided in Sec. VI. Simulation experiments are provided in Sec. VII. We conclude in Sec. VIII.

II. Problem Set-up, Notation, and Geometric Median Preliminaries

A. Problem Setup

We study two interrelated problems stated below. We begin by stating the subspace estimation meta problem that occurs in both problems. We consider a federated setting with L nodes, with L being a numerical constant, and assume the following.

Assumption 1 (Number of Byzantine Nodes): At most τL of the L total nodes are Byzantine, with $\tau \leq 0.4$ (instead of 0.4, we can use any constant c that is strictly less than 0.5 here). Denote the set of good (non-Byzantine) nodes by $\mathcal{J}_{good}$. Equivalently, this means that $|\mathcal{J}_{good}| > (1-\tau)L$. We define a Byzantine attack below in Sec. II-A.4.

1) Resilient Federated Subspace Estimation: The goal is to reliably estimate the subspace spanned by the top r singular vectors of an unknown symmetric $n \times n$ matrix, $\boldsymbol{\Phi}^*$. Denote the $n \times r$ matrix formed by these singular vectors by $\boldsymbol{U}^*$. Our goal is thus to estimate $\mathrm{span}(\boldsymbol{U}^*)$. Each node $\ell \in [L]$ observes, or can compute, a symmetric matrix $\boldsymbol{\Phi}_\ell$ which is *an* estimate of $\boldsymbol{\Phi}^*$. Typically, the node observes an $n \times q_\ell$ data matrix $\boldsymbol{D}_\ell$ and computes $\boldsymbol{\Phi}_\ell := \boldsymbol{D}_\ell \boldsymbol{D}_\ell^\top / q_\ell$. We use $\sigma_1^* \geq \ldots \geq \sigma_n^*$ to denote the singular values of $\boldsymbol{\Phi}^*$.

2) Resilient Federated PCA: Given q data vectors $\boldsymbol{d}_k \in \Re^n$, that are mutually independent and identically distributed (i.i.d.), the goal is to find the r-dimensional principal subspace (span of top r singular vectors) of their covariance matrix, which we will denote by $\boldsymbol{\Sigma}^*$. We can arrange the data vectors into an $n \times q$ matrix, $\boldsymbol{D} := [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots \boldsymbol{d}_q]$. We use σ_j^* to denote the j-th singular value of $\boldsymbol{\Sigma}^*$. We assume that all $\boldsymbol{d}_k$s are i.i.d. *zero mean*, sub-Gaussian vectors, with covariance

[2]As explained earlier, the conditional expectations are different at the different nodes. These can be computed and bounded easily because we assume sample-splitting.

matrix $\mathbf{\Sigma}^*$ and maximum sub-Gaussian norm $K\sqrt{\|\mathbf{\Sigma}^*\|} = K\sqrt{\sigma_1^*}$ [56, Chap 2]. The data is vertically federated, this means that each node ℓ has $q_\ell = \widetilde{q} = \frac{q}{L}$ $\boldsymbol{d}_k$'s. Denote the corresponding sub-matrix of $\boldsymbol{D}$ by $\boldsymbol{D}_\ell$. Thus, $\boldsymbol{D} = [\boldsymbol{D}_1, \ldots, \boldsymbol{D}_\ell, \ldots \boldsymbol{D}_L]$. This problem is an instance of resilient federated subspace estimation with $\mathbf{\Phi}_\ell = \boldsymbol{D}_\ell \boldsymbol{D}_\ell^\top / q_\ell$.

3) Resilient Horizontally-Federated Low Rank Column-Wise Sensing (LRCS): LRCS involves recovering an $n \times q$ rank-r matrix $\boldsymbol{X}^*$ from compressive linear measurements of each column, i.e., from $\boldsymbol{y}_k := \boldsymbol{A}_k \boldsymbol{x}_k^*$, $k \in [q]$, with $\boldsymbol{y}_k \in \Re^m$ with $m \ll n$, and $\boldsymbol{A}_k$ being $m \times n$ matrices which are i.i.d. random Gaussian (each entry is i.i.d. standard Gaussian) [49]. We treat $\boldsymbol{X}^*$ as a deterministic unknown. Here and below $[q] := \{1, 2, \ldots, q\}$. Let $\boldsymbol{Y} := [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_q]$. Horizontal federation means that row sub-matrices of $\boldsymbol{Y}$ are observed at the different nodes. To be precise, node ℓ observes an $\widetilde{m} \times q$ rows sub-matrix of $\boldsymbol{Y}$ denoted $\boldsymbol{Y}_\ell$ with $\widetilde{m} = m/L$. We assume that node ℓ has access to $\boldsymbol{Y}_\ell$ and $\{(A_k)_\ell, k \in [q]\}$. Denote the set of indices of the rows available at node ℓ by $\mathcal{S}_\ell$. Then, $(\boldsymbol{A}_k)_\ell := \boldsymbol{I}_{\mathcal{S}_\ell}^\top \boldsymbol{A}_k$ is of size $\widetilde{m} \times n$ and $\boldsymbol{Y}_\ell := \boldsymbol{I}_{\mathcal{S}_\ell}^\top \boldsymbol{Y}$ is of size $\widetilde{m} \times q$ with $\widetilde{m} = m/L$, and with $(\boldsymbol{y}_k)_\ell := (\boldsymbol{A}_k)_\ell \boldsymbol{x}_k^*$ for all $k \in [q]$ Three important applications that can be modeled as instances of LRCS are accelerated dynamic MRI reconstruction [9], [10], federated sketching [10], [15], and multi-task representation learning and few shot learning [2], [11], [12]. In the representation learning problem, horizontal federation corresponds to the setting where the ℓ-th subset of training data for the q correlated linear regression tasks is observed at node ℓ. Few shot learning uses this learned representation (column span of $\boldsymbol{X}^*$) for learning the regression coefficients using very few training data points (this problem is also referred to as online subspace tracking in [10]). In multi-coil dynamic MRI, L is the number of MRI scanners, each of which observes a differently weighted subset of measurements of the human organ's image sequence. Scanners can be prone to security threats if they are connected to the internet. As we will see later, the initialization step for solving LRCS using an iterative algorithm can be interpreted as an instance of resilient federated subspace estimation.

The reason we consider vertical federation for PCA but horizontal for LRCS is because these are the settings in which the data on the different nodes is i.i.d. in each case. In case of vertically federated PCA, $\boldsymbol{D}_\ell$'s are i.i.d. If we consider horizontal federation for PCA, then this is no longer true (unless we assume $\mathbf{\Sigma}^*$ is block diagonal). For LRCS, the opposite holds because different entries of a given $\boldsymbol{y}_k$ are i.i.d.; but the different $\boldsymbol{y}_k$'s are not identically distributed. Guaranteeing Byzantine resilience without extra assumptions requires the different nodes' data be i.i.d. or i.i.d.-like (this means that it should be possible to obtain a uniform bound on the errors between the individual nodes' outputs and the quantity of interest each time the node output is shared with the center). As we explain later, it is possible to use ideas similar to the ones introduced here to also solve vertically federated LRCS, but that will need extra assumptions that ensure bounded heterogeneity.

4) Byzantine Attack Definition: We use the terms Byzantine node/adversary/attack almost interchangeably. The output of a Byzantine node or adversary is the Byzantine attack. Byzantine nodes are often also referred to as "bad" nodes and non-Byzantine ones as "good" nodes. The Byzantine attack has not been clearly mathematically defined in past work [4], [5], [6], [7], [8], although there are definitions inspired by [57]. The following definition, taken from [4], is the most precise one we can find.

Definition 1: The Byzantine adversary is an entity which controls the outputs of some of the L worker nodes. It is omniscient, in the sense that it has a perfect knowledge of the system state at any time, i.e., it knows (i) the full state of the center (data and algorithm, including all algorithm parameters), and (ii) the full state of every node (data and algorithm, including all algorithm parameters). Different Byzantine adversaries can also collude. However, they are not omnipotent: they cannot modify the outputs of the other (non-Byzantine) nodes or of the center, or delay communication.

In our setting, this means the following. Let ∇_{byz} denote the set of outputs of all the Byzantine nodes. Then $\nabla_{byz} = g_{byz}(\{\mathrm{Data}_\ell\}_{\ell=1}^L, \mathcal{A}))$ where $\mathcal{A}$ denotes the true algorithm being implemented at each of the non-Byzantine (good) nodes and at the center along with all its parameters; $g_{byz}(.)$ is a function that can be jointly designed by all the Byzantine nodes; and Data_ℓ is the data observed at node ℓ: it is $\mathbf{\Phi}_\ell$ or $\boldsymbol{D}_\ell$ (in case of PCA), or $\boldsymbol{Y}_\ell, (\boldsymbol{A}_k)_\ell, k \in [q]$ (in case of LRCS).

B. Notation

We use $\|.\|_F$ to denote the Frobenius norm and $\|.\|$ without a subscript to denote the (induced) l_2 norm (often called the operator norm or spectral norm); $^\top$ denotes matrix or vector transpose; $|\boldsymbol{z}|$ for a vector $\boldsymbol{z}$ denotes element-wise absolute values; $\boldsymbol{I}_n$ (or sometimes just $\boldsymbol{I}$) denotes the $n \times n$ identity matrix, and $\boldsymbol{e}_k$ denotes its k-th column (k-th canonical basis vector); and $\boldsymbol{M}^\dagger = (\boldsymbol{M}^\top \boldsymbol{M})^{-1} \boldsymbol{M}^\top$. We use $\mathbb{1}_{\{a \le b\}}$ to denote the indicator function that returns 1 if $a \le b$ otherwise 0.

We say $\boldsymbol{U}$ is a *basis matrix* if it is a tall matrix with mutually orthonormal columns; we use this to denote the subspace spanned by its columns. For a basis matrix $\boldsymbol{U}$, the projection matrix for projecting onto $\mathrm{span}(\boldsymbol{U})$ (the subspace spanned by the columns of $\boldsymbol{U}$) is denoted $\mathcal{P}_{\boldsymbol{U}} := \boldsymbol{U}\boldsymbol{U}^\top$ while that for projecting orthogonal to $\mathrm{span}(\boldsymbol{U})$ is denoted $\mathcal{P}_{\boldsymbol{U},\perp} := \boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top$ "r-SVD" to refer to the top r left singular vectors (singular vectors corresponding to the r largest singular values) of a matrix. For basis matrices, $\boldsymbol{U}_1, \boldsymbol{U}_2$, we use $\boldsymbol{SD}_F(\boldsymbol{U}_1, \boldsymbol{U}_2) := \|(\boldsymbol{I} - \boldsymbol{U}_1\boldsymbol{U}_1^\top)\boldsymbol{U}_2\|_F$ as the default Subspace Distance (SD) measure between the subspaces spanned by the two matrices. In some places, we also use $\boldsymbol{SD}_2(\boldsymbol{U}_1, \boldsymbol{U}_2) = \|(\boldsymbol{I} - \boldsymbol{U}_1\boldsymbol{U}_1^\top)\boldsymbol{U}_2\|$. If both matrices have r columns (denote r-dimensional subspaces), then $\boldsymbol{SD}_F(\boldsymbol{U}_1, \boldsymbol{U}_2) \le \sqrt{r}\boldsymbol{SD}_2(\boldsymbol{U}_1, \boldsymbol{U}_2)$.

We use $QR(\tilde{\boldsymbol{U}})$ to denote the orthonormalization of the columns of $\tilde{\boldsymbol{U}}$ by using QR decomposition. For a matrix $\boldsymbol{M}$, $vec(\boldsymbol{M})$ vectorizes it.

We reuse the letters c, C to denote different numerical constants in each use with the convention that $c < 1$ and $C \ge 1$. Also the notation $a \lesssim b$ means $a \le Cb$.

For a, b in $(0,1)$, we use $\psi(a,b) := (1-a)\log\frac{1-a}{1-b} + a\log\frac{a}{b}$ to denote the binary KL divergence. When computing a median of means estimator, one splits the L node indices into $\tilde{L}$ mini-batches so that each mini-batch contains $\rho = L/\tilde{L}$ indices. For the ℓ-th node in the ϑ-th mini-batch we use the short form notation $(\vartheta,\ell) = (\vartheta-1)\rho + \ell$, for $\ell \in [\rho]$.

Recall that Byzantine nodes are often also referred to as "bad" nodes and non-Byzantine ones as "good" nodes. We use $\mathcal{J}_{good} \subseteq [L]$ to denote the set of non-Byzantine/good nodes and $\mathcal{J}_{good}^c$ denotes the set of Byzantine nodes. Using Assumption 1, $|\mathcal{J}_{good}| = L - \tau L = (1-\tau)L$.

C. Geometric Median (GM)

The geometric median (GM) is one well-known approach to compute a reliable estimate of a vector-valued quantity using multiple individual estimates of it when some of these estimates may be corrupted by outliers [16], [46]. For L data vectors $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_L\}$, with each $\boldsymbol{z}_\ell \in \Re^d$, this is defined as

$$\boldsymbol{z}_{GM}^* := GM\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_L\} = \arg\min_{\boldsymbol{z}\in\Re^d} \sum_{\ell=1}^{L} \|\boldsymbol{z} - \boldsymbol{z}_\ell\|$$

This cannot be computed in closed form. Iterative algorithms exist to solve it approximately. When we say $\boldsymbol{z}_{GM}$ is a $(1+\epsilon_0)$ approximate GM, for an $0 < \epsilon_0 < 1$ we mean that

$$\begin{aligned}\sum_{\ell=1}^{L} \|\boldsymbol{z}_{GM} - \boldsymbol{z}_\ell\| &\le (1+\epsilon_0) \sum_{\ell=1}^{L} \|\boldsymbol{z}_{GM}^* - \boldsymbol{z}_\ell\| \\ &= (1+\epsilon_0) \min_{\boldsymbol{z}\in\Re^d} \sum_{\ell=1}^{L} \|\boldsymbol{z} - \boldsymbol{z}_\ell\| \qquad (1)\end{aligned}$$

There are two popular iterative solutions for computing the approximate GM. The most commonly used one in practice, Weiszfeld's algorithm [58], [59], does not come with a useful iteration complexity guarantee. The recent work of [60] introduced a nearly linear-time algorithm for provably computing the approximate GM, with high probability. We provide [60, Algorithm 1] in Appendix D. We state its guarantee next. All theoretical results in our work use this result.

Claim 1 (Theorem 1 [60]): Pick an accuracy level $0 < \epsilon_0 < 1$. Consider [60, Algorithm 1] with input $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_L\}$ and using number of iterations, $T_{GM} = C\log(\frac{L}{\epsilon_0})$. With probability at least $1 - c_{approxGM}$ (where $c_{approxGM} < 1$ is a numerical constant, e.g., 0.1), the algorithm computes $\boldsymbol{z}_{GM}$ that satisfies (1). Its per iteration complexity is $CLd\log^2(\frac{L}{\epsilon_0})$ and total time complexity is $O(Ld\log^3(\frac{L}{\epsilon_0}))$.

The use of the above result allows us to bound the iteration complexity of all our algorithms. This, in turns, allows us to get a bound on the total communication cost and the total time cost. Although it has a simple guarantee, the algorithm [60, Algorithm 1] itself is quite complicated. The authors of [60] have not shown any experimental results with it. To our best knowledge, nor have any other authors in follow-up work that cites it. The algorithm used in practice for approximating the GM is the Weiszfeld's algorithm initialized using the average of the $\boldsymbol{z}_\ell$'s [58]. This is an iteratively re-weighted least squares type algorithm. We provide both algorithms in Appendix D.

Algorithm 1 Subspace Median

Input $\boldsymbol{D}_\ell$, $\ell \in [L]$; or $\boldsymbol{\Phi}_\ell$, $\ell \in [L]$.
Parameters T_{GM}, T_{pow}
1: **Nodes** $\ell = 1, \ldots, L$
2: Compute top r singular vectors, $\hat{\boldsymbol{U}}_\ell$, of $\boldsymbol{D}_\ell$ (equivalently of $\boldsymbol{\Phi}_\ell := \boldsymbol{D}_\ell\boldsymbol{D}_\ell^\top$).
(Can use power method with T_{pow} iterations)
3: **Central Server**
4: Orthonormalize: $\boldsymbol{U}_\ell \leftarrow QR(\hat{\boldsymbol{U}}_\ell)$, $\ell \in [L]$
5: Compute Projection Matrices: $\mathcal{P}_{\boldsymbol{U}_\ell} \leftarrow \boldsymbol{U}_\ell\boldsymbol{U}_\ell^\top$, $\ell \in [L]$
6: Compute their GM: $\mathcal{P}_{GM} \leftarrow approxGM\{\mathcal{P}_{\boldsymbol{U}_\ell},\ \ell \in [L]\}$
(Use [60, Algorithm 1] with parameter T_{GM}).
7: Find $\ell_{best} = \arg\min_\ell \|\mathcal{P}_{\boldsymbol{U}_\ell} - \mathcal{P}_{GM}\|_F$
8: Output $\boldsymbol{U}_{out} = \boldsymbol{U}_{\ell_{best}}$

III. Resilient Federated Subspace Estimation

A. Proposed Solution: Subspace Median

Recall that our goal is to obtain a reliably accurate estimate of $\mathrm{span}(\boldsymbol{U}^*)$, which is an r-dimensional subspace in $\Re^n$, when each node computes an estimate $\boldsymbol{U}_\ell$ of it by computing the top r singular vectors of $\boldsymbol{\Phi}_\ell$.. Some nodes can be Byzantine (Assumption 1). We develop a solution approach that relies on the geometric median (GM). Notice from Sec. II-C that the GM is defined for quantities whose distance can be measured using the vector l_2 norm (equivalently, matrix Frobenius norm). Our solution adapts the GM to use it for subspaces by using the fact that the Frobenius norm between the projection matrices of two subspaces is another measure of subspace distance: $\|\mathcal{P}_{\boldsymbol{U}} - \mathcal{P}_{\boldsymbol{U}^*}\|_F = \sqrt{2}\boldsymbol{SD}_F(\boldsymbol{U}, \boldsymbol{U}^*)$ [61, Lemma 2.5].

Our proposed algorithm, which we refer to as "Subspace Median", relies on this fact. It proceeds as follows. Each node computes the top r singular vectors of its matrix $\boldsymbol{\Phi}_\ell$, denoted $\hat{\boldsymbol{U}}_\ell$, and sends these to the center. If node ℓ is good (non-Byzantine), then $\hat{\boldsymbol{U}}_\ell$ already has orthonormal columns; however if the node is Byzantine, then it is not. The center first orthonormalizes the columns of all the received $\hat{\boldsymbol{U}}_\ell$'s using QR. This ensures that all the $\boldsymbol{U}_\ell$'s have orthonormal columns. It then computes the projection matrices $\mathcal{P}_{\boldsymbol{U}_\ell} := \boldsymbol{U}_\ell\boldsymbol{U}_\ell^\top$, $\ell \in [L]$, followed by vectorizing them, computing their GM, and converting the GM back to an $n \times n$ matrix. Denote this by $\mathcal{P}_{GM}$. Finally, the center finds the ℓ for which $\mathcal{P}_{\boldsymbol{U}_\ell}$ is closest to $\mathcal{P}_{GM}$ in Frobenius norm and outputs the corresponding $\boldsymbol{U}_\ell$.

We should mention that this last step can also be replaced by finding the top r singular vectors of $\mathcal{P}_{GM}$. However, doing this requires time of order $n^2 r\log(1/\epsilon)$ while finding the closest $\mathcal{P}_{\boldsymbol{U}_\ell}$ only needs time of order $\max(n^2, L\log L)$.

Subspace Median is summarized in Algorithm 1. We can prove the following for it.

Lemma 1 (Subspace-Median): For a $\delta > 0$, consider Algorithm 1 with $T_{GM} = C\log\left(\frac{Lr}{\delta}\right)$. Assume that Assumption 1 holds. Assume that, for at least $(1-\tau)L$ nodes, the following holds:

$$\Pr\left(\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_\ell) \le \delta\right) \ge 1 - p.$$

Then, w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4-\tau,p))$,

$$\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_{out}) \leq 23\delta.$$

Here $\psi(a,b) := (1-a)\log\frac{1-a}{1-b} + a\log\frac{a}{b}$ for $0 < a, b < 1$ is the binary KL divergence, and $c_{approxGM}$ is the numerical constant from Claim 1.

Proof: See Sec. VI. □

Combining this lemma with the Davis-Khan $\sin\Theta$ theorem (bounds the distance between the principal subspaces of two symmetric matrices) [62] and a guarantee for the power method [63], we can prove the following theorem.

Remark 1: We specify the power method just to have one algorithm for computing the top r singular vectors of a matrix for which we can specify the time compleixty. It can be replaced by any other algorithm and our overall result remains the same.

Theorem 3.1 (Subspace-Median Guarantee): Pick an $\epsilon < 1$. Assume that Assumption 1 holds and that $\sigma_r^* - \sigma_{r+1}^* \geq \Delta$ for a $\Delta > 0$. Assume also that, for at least $(1-\tau)L$ node outputs, the following holds, for a $p > 0$.

$$\Pr\left\{\|\boldsymbol{\Phi}_\ell - \boldsymbol{\Phi}^*\| \leq \frac{\epsilon}{92\sqrt{r}}\Delta\right\} \geq 1 - p.$$

Consider Algorithm 1 with $T_{GM} = C\log\left(\frac{Lr}{\epsilon}\right)$.

1) Assume use of exact SVD at the nodes. Then, w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4-\tau,p))$,

$$\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \leq \epsilon$$

2) Assume that the power method with T_{pow} iterations is used for the SVD step. If $T_{pow} = C\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$, then the above conclusion holds w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4-\tau, p+\frac{1}{n^{10}}))$.
The communication cost is nr per node. The computational cost at the center is order $n^2 L\log^3\left(\frac{Lr}{\epsilon}\right)$. The computational cost at any node (when using power method) is order $nq_\ell r T_{pow} = nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$.

Proof: See Sec. VI. □

The assumption $\sigma_r^* - \sigma_{r+1}^* \geq \Delta$ (singular value gap) is needed for ensuring that the span of $\boldsymbol{U}_\ell$ computed at any good node is an accurate estimate of the span of $\boldsymbol{U}^*$. It also decides the time complexity of the computation (Δ appears in the required number of power method iterations.

B. Alternate Solution 1: SVD on Resilient Covariance Estimation (SVD-ResCovEst)

SVD-ResCovEst is the solution studied by Minsker [46] and described earlier. It involves computing the GM of (vectorized) $\boldsymbol{\Phi}_\ell$s, followed by obtaining the principal subspace (r-SVD) of the GM matrix. In a federated setting, this is communication inefficient since it requires that each node ℓ either shares its raw data $\boldsymbol{D}_\ell$ with the center (this is a matrix of size $n \times q_\ell$), or, that it shares $\boldsymbol{\Phi}_\ell = \boldsymbol{D}_\ell\boldsymbol{D}_\ell^\top/q_\ell$ (this is of size $n \times n$). For PCA, as we explain in the next section, this is also sample inefficient; it requires $q_\ell \geq n^2/\epsilon^2$. See Remark 3.

Algorithm 2 Resilient Power Method (ResPowMeth)

Parameters $T_{pow}, T_{GM}, \omega_{GM}$

1: **Central Server** Randomly Initialize $\boldsymbol{U}_{rand}$ with i.i.d standard Gaussian entries. Set $\boldsymbol{U}_0 = \boldsymbol{U}_{rand}$.
2: **for** $t \in T_{pow}$ **do**
3: **Nodes** $\ell = 1, \ldots, L$
4: Compute $\boldsymbol{\Phi}_\ell\boldsymbol{U}_{t-1}$
5: **Central Server**
6: $GM \leftarrow approxGM\left(\{vec(\boldsymbol{\Phi}_\ell\boldsymbol{U}_{t-1}),\ \ell \in [L]\}\backslash \{\ell : \|\boldsymbol{\Phi}_\ell\boldsymbol{U}_{t-1}\|_F > \omega_{GM}\}\right)$
(Use [60, Algorithm 1] with T_{GM} iterations on the set of $\boldsymbol{\Phi}_\ell\boldsymbol{U}_{t-1}$s whose Frobenius norm is below ω_{GM})
7: Orthonormalize: using QR $GM \stackrel{QR}{=} \hat{\boldsymbol{U}}\boldsymbol{R}$
8: Return $\boldsymbol{U}_t \leftarrow \hat{\boldsymbol{U}}$
9: **end for**
10: Output $\boldsymbol{U}_{out} \leftarrow \boldsymbol{U}_{T_{pow}}$

C. Alternate Solution 2: Resilient Power Method (ResPowMeth)

A natural way to make the SVD-ResCovEst approach communication-efficient is to borrow ideas from the sketching literature and share $\boldsymbol{\Phi}_\ell\boldsymbol{U}_{any}$ for some (possibly random) $n \times r$ matrix $\boldsymbol{U}_{any}$. This idea is, in fact, one iteration of the power method for computing the r-SVD of a matrix [54], [55]. It can be converted into a provably correct solution by using the GM to modify the power method. This starts with a random Gaussian initialization, $\boldsymbol{U}_{rand}$, and implements the iteration: $\boldsymbol{U} \leftarrow \mathrm{QR}(\sum_\ell \boldsymbol{\Phi}_\ell\boldsymbol{U})$. In our GM based modification, we replace the summation by the GM. We refer to this solution as *Resilient Power Method (ResPowMeth)*, and summarize it in Algorithm 2. As we show next, ResPowMeth works with high probability (w.h.p.) if all the $\boldsymbol{\Phi}_\ell$'s are very accurate estimates of $\boldsymbol{\Phi}^*$. The reason it needs to make the above assumption is because it computes the GM of the node outputs $\boldsymbol{\Phi}_\ell\boldsymbol{U}$ at each iteration including the first one. At the first iteration, $\boldsymbol{U}_0$ is a randomly generated matrix and thus, w.h.p., this is a bad approximation of the desired subspace $\mathrm{span}(\boldsymbol{U}^*)$. Consequently, the same is true for the column span of $\tilde{\boldsymbol{U}}_\ell^+ = \boldsymbol{\Phi}_\ell\boldsymbol{U}_0$. To understand this easily, suppose $\boldsymbol{U}_0$ is almost orthogonal to $\boldsymbol{U}^*$, i.e., $\boldsymbol{U}_0^\top\boldsymbol{U}^* \approx \boldsymbol{0}$. Then the span of $\tilde{\boldsymbol{U}}_\ell^+$ will be almost orthonormal to that of $\boldsymbol{U}^*$. Thus, unless all the Φ_ℓs are very similar, the column spans of the different $\tilde{\boldsymbol{U}}_\ell^+$s will not be close. As a result, the GM of their projection matrices will not be able to distinguish between the good and Byzantine ones. There is a good chance that it approximates the subspace of the Byzantine one(s). This then means that the updated $\boldsymbol{U}$ is also a bad approximation of $\mathrm{span}(\boldsymbol{U}^*)$. The same idea repeats at the second iteration. Thus, with significant probability, the subspace estimates do not improve over iterations. This intuition is captured in the guarantee provided next. It becomes clearer in the direct one-step analysis that we provide in Appendix C.

Theorem 3.2 (ResPowMeth Guarantee): Assume that Assumption 1 holds and that $\sigma_r^* - \sigma_{r+1}^* \geq \Delta$ for a

$\Delta > 0$. Consider ResPowMeth (Algorithm 2) with $T_{pow} = C\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$, $T_{GM} = \log\left(\frac{Lnr}{\epsilon}\right)$, and $\omega_{GM} = 1.1\sigma_1^*\sqrt{r}$. Suppose, for at least $(1-\tau)L$ node outputs, the following holds

$$\Pr\left\{\|\mathbf{\Phi}_\ell - \mathbf{\Phi}^*\| \le \frac{1}{70}\min\left(\frac{\epsilon}{\sqrt{r}}, \frac{1}{2\sqrt{nr}}\right)\Delta\right\} \ge 1-p.$$

Then w.p. at least $1 - c_{approxGM} - c - Lp - \exp(-L\psi(0.4-\tau, p))$ $\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \le \epsilon$. The communication cost is $nrT_{pow} = Cnr\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$ per node. The computational cost at the center is $nrL\log^3\left(\frac{Lnr}{\epsilon}\right)\cdot T_{pow} = nrL\frac{\sigma_r^*}{\Delta}\log^3\left(\frac{Lnr}{\epsilon}\right)\log(\frac{n}{\epsilon})$. The computational cost at any node is $nq_\ell rT_{pow} = nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$.

Proof: See Sec. VI. A second more illustrative proof is provided in Appendix C. □

Observe that this result assumes $\|\mathbf{\Phi}_\ell - \mathbf{\Phi}^*\| \lesssim \min(\epsilon/\sqrt{r}, 1/(\sqrt{n}\,r))\cdot\Delta$. The $1/\sqrt{nr}$ factor makes this a very stringent requirement, e.g., even to get an $\epsilon = 0.5$ accurate subspace estimate, we need $\|\mathbf{\Phi}_\ell - \mathbf{\Phi}^*\| \lesssim \Delta/\sqrt{n}\,r$. On the other hand, the Subspace Median guarantee only assumes $\|\mathbf{\Phi}_\ell - \mathbf{\Phi}^*\| \lesssim (\epsilon/\sqrt{r})\Delta$. As we will see in the next section, this translates into a much better sample complexity for PCA for Subspace Median than for ResPowMeth.

IV. Application 1: Resilient Federated PCA

Recall from Sec. II-A that our goal is to reliably estimate the principal subspace of the unknown data covariance matrix $\mathbf{\Sigma}^*$. Node ℓ has access to a subset of q_ℓ data vectors $\boldsymbol{d}_k$ arranged as columns of an $n \times q_\ell$ matrix $\boldsymbol{D}_\ell$.

A. Subspace-Median (SubsMed) for Resilient PCA

Using its data, each node can compute the empirical covariance matrix $\hat{\mathbf{\Sigma}}_\ell := \boldsymbol{D}_\ell\boldsymbol{D}_\ell^\top/\widetilde{q}$. This is an estimate of the true one, $\mathbf{\Sigma}^*$. This allows us to use Algorithm 1 applied to $\boldsymbol{D}_\ell$ or $\hat{\mathbf{\Sigma}}_\ell$ to obtain a Byzantine resilient PCA solution, and use Theorem 3.1 to analyze it. The sample complexity needed to get the desired bound on $\|\hat{\mathbf{\Sigma}}_\ell - \mathbf{\Sigma}^*\|$ w.h.p. is obtained using [56, Theorem 4.7.1]. Combining these two results, we can prove the following.

Corollary 1 (Subspace Median for PCA): Consider the PCA problem as defined in Sec. II-A.2. Assume that Assumption 1 holds and that $\sigma_r^* - \sigma_{r+1}^* \ge \Delta$ for a $\Delta > 0$. Consider Algorithm 1 (SubsMed) with input $\mathbf{\Phi} = \boldsymbol{D}_\ell\boldsymbol{D}_\ell^\top/q_\ell$, and parameters as set in Theorem 3.1. If

$$q_\ell := \frac{q}{L} \ge CK^4\frac{\sigma_1^{*2}}{\Delta^2}\cdot\frac{nr}{\epsilon^2},$$

then, w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4-\tau, 2\exp(-n)+n^{-10}))$, $\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \le \epsilon$.

Proof: We prove it in Sec. VI-F. It is an immediate corollary of Theorem 3.1 and [56, Theorem 4.7.1]. □

Remark 2 (ResPowMeth for PCA): In the setting of Corollary 1, consider Algorithm 2 (ResPowMeth). If $q_\ell \ge CK^4\frac{\sigma_1^{*2}}{\Delta^2}\cdot\max\left(\frac{n}{\epsilon^2}, n^2r^2\right)$, then $\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \le \epsilon$.

Remark 3 (SVD-CovEst for PCA): In the setting of Corollary 1, consider SVD-ResCovEst (SVD on GM of nodes' covariance matrix estimates) studied in [46, Corollary 4.3]. By using [46, Corollary 4.3], and using the fact that, $\mathbb{E}\|\boldsymbol{d}_k\|^4 - \text{trace}(\Sigma^{*2}) \le Cn^2K^4\sigma_1^{*2}$ under the sub-Gaussian assumption, we can conclude the following: If $q_\ell \ge CK^4\frac{\sigma_1^{*2}}{\Delta^2}n^2/\epsilon^2$, then, $\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \le \epsilon$ with constant probability.

The reason this needs q_ℓ of order n^2 is because it first obtains a resilient estimate of the entire $n \times n$ covariance matrix, followed by r-SVD on it. For resilient estimation, it needs to use the Frobenius norm as the error measure. The robust estimator studied in [42, Theorem 4.35] uses a different algorithm, but this also needs order n^2/ϵ^2 sample complexity and order n^2 communication complexity.

Remark 4 (Generalizations of Theorem 3.1): (1) Theorem 3.1 also holds if the $\boldsymbol{d}_k$'s are not i.i.d., but are zero mean, independent, sub-Gaussian, and with covariance matrices that are of the form $\mathbb{E}[\boldsymbol{d}_k\boldsymbol{d}_k^\top] = \boldsymbol{U}^*\boldsymbol{S}_k\boldsymbol{U}^{*\top}$ with all $\boldsymbol{S}_k$'s being such that their r-th singular value gap is at least Δ.

(2) We can also combine Theorem 3.1 with the sample complexity bound for estimating approximately LR covariance matrices given in [64, Corollary 5.52 and Remark 5.53] to show that, in this case, a much lower sample complexity suffices. Suppose $\boldsymbol{d}_k$ are i.i.d., *zero mean*, sub-Gaussian, have covariance matrix $\mathbf{\Sigma}^*$, and are bounded with $\|\boldsymbol{d}_k\|^2 \le K^2 trace(\mathbf{\Sigma}^*)$ and $trace(\mathbf{\Sigma}^*) = r_0\sigma_1^*$ with stable rank $r_0 \ll n$ (approximately LR matrix). Then, if $q_\ell \ge CK^4\frac{\sigma_1^{*2}}{\Delta^2}(\max(r_0, r)^2\log n)/\epsilon^2$, then $\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \le \epsilon$. Here r_0 is the stable rank.

(3) We can also do the above for PCA with missing data by combining with [65, Theorem 3.22].

B. Subspace Median-of-Means (Subspace MoM)

As is well known, the use of median of means (MoM), instead of median, improves (reduces) the sample complexity needed to achieve a certain recovery error, but tolerates a smaller fraction of Byzantine nodes. It is thus useful in settings where the number of bad nodes is small. We show next how to obtain a communication-efficient and private GMoM estimator for federated PCA. Pick an integer $\tilde{L} \le L$. In order to implement the "mean" step, we need to combine samples from $\rho = L/\tilde{L}$ nodes, i.e., we need to find the r-SVD of matrices $\boldsymbol{D}_{(\vartheta)} = [\boldsymbol{D}_{(\vartheta,1)}, \boldsymbol{D}_{(\vartheta,2)}, \ldots, \boldsymbol{D}_{(\vartheta,\rho)}]$, for all $\vartheta \in [\tilde{L}]$. Recall that $_{(\vartheta,\ell)} =_{(\vartheta-1)\rho+\ell}$. This needs to be done without sharing the entire matrix $\boldsymbol{D}_{(\vartheta,1)}$. We do this by implementing $\tilde{L}$ different federated power methods, each of which combines samples from a different minibatch of ρ nodes. The output of this step will be $\tilde{L}$ subspace estimates $\boldsymbol{U}_{(\vartheta)}$, $\vartheta \in [\tilde{L}]$. These serve as inputs to the Subspace-Median algorithm to obtain the final Subspace-MoM estimator. We summarize the complete the algorithm in Algorithm 3. We should mention that $\tilde{L} = L$ is the subspace median special case.

As long as the same set of τL nodes are Byzantine for all the power method iterations, we can prove the following.

Corollary 2: Consider Algorithm 3 and the setting of Corollary 1. Assume that the set of Byzantine nodes remains fixed for all iterations in this algorithm and the size of this set is at most τL with $\tau < 0.4\tilde{L}/L$. If

$$\frac{q}{L} = \widetilde{q} \ge CK^4\frac{\sigma_1^{*2}}{\Delta^2}\frac{nr}{\epsilon^2}\cdot\frac{\tilde{L}}{L}$$

TABLE I

COMPARING SUBSPACE MEDIAN (SUBSMED) WITH SVD-RESCOVEST AND RESILIENT POWER METHOD (RESPOWMETH) AND WITH THE BASIC POWER METHOD FOR A NO-ATTACK SETTING. WE OBTAIN COMPLEXITIES FOR GUARANTEEING $\boldsymbol{SD}_2(\boldsymbol{U},\boldsymbol{U}^*) \leq \epsilon$. SUBSMED AND RESPOWMETH ONLY BOUND $\boldsymbol{SD}_F(\boldsymbol{U},\boldsymbol{U}^*)$ AND THIS IS WHY THE SAMPLE COMPLEXITIES FOR THESE CONTAIN AN EXTRA FACTOR OF r. THIS TABLE SUMMARIZES THE RESULTS OF COROLLARY 1 AND THE TWO REMARKS BELOW IT

Methods→	**SVD-ResCovEst** [46, Cor 4.3], [42, Thm 4.35]	**ResPowMeth (Proposed modific of [46, Cor 4.3])**	**SubsMed (Proposed)**	**Basic PowMeth, no attack** (Baseline)
Sample Comp for PCA $q \geq CK^4 \frac{{\sigma_1^*}^2}{\Delta^2} \times$	$\frac{n^2}{\epsilon^2} \cdot L$	$\max\left(n^2 r^2, \frac{n}{\epsilon^2}\right) \cdot L$	$\frac{nr}{\epsilon^2} \cdot L$	$\frac{n}{\epsilon^2}$
Communic Cost	n^2	$nr\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$	nr	$nr\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$
Compute Cost - node	$n^2 q_\ell$	$nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$	$nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$	$nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$
Compute Cost - center	$n^2 L\log^3\left(\frac{Lr}{\epsilon}\right)$	$\frac{\sigma_r^*}{\Delta}nrL\log(\frac{n}{\epsilon})\log^3\left(\frac{Ln}{\epsilon}\right)$	$n^2 L\log^3\left(\frac{Lr}{\epsilon}\right)$	$\frac{\sigma_r^*}{\Delta}nrL\log(\frac{n}{\epsilon})$

then, the conclusion of Corollary 1 holds. The communication cost is $T_{pow}nr = nr\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$ per node. The computational cost at the center is order $n^2\tilde{L}\log^3\left(\frac{\tilde{L}r}{\epsilon}\right)$. The computational cost at any node is order $nq_\ell r\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$.

C. Discussion and Comparisons

1) Comparing Subspace-Median and Subspace-MoM: For a chosen value of $\tilde{L} \leq L$, the sample complexity required by subspace-MoM reduces by a factor of $1/\rho = \tilde{L}/L$, but its Byzantine tolerance also reduces by this factor. This matches what is well known for other MoM estimators, e.g., that for gradients used in [16]. Also, the communication cost of Subspace-MoM is larger than that of Subspace Median since it implements a power method to share samples between subsets of nodes.

2) Comparing Subspace-Median With SVD-ResCovEst and ResPowMeth: Consider communication cost. SVD-CovEst has a very high cost of order n^2 while Subspace Median and ResPowMeth have much lower costs of order nr and $nr\frac{\sigma_r^*}{\Delta}$ times a log factor respectively. Consider sample cost. Both SVD-CovEst and ResPowMeth have a very high sample cost of order n^2 and order n^2r^2 respectively for $\epsilon = c$. Subspace Median has a sample cost of only order nrL.

In terms of computation cost at the nodes, SVD-CovEst is the most expensive, while both ResPowMeth and Subspace Median have the same cost. But, at the center, Subspace Median has a higher cost by a factor $n/(r\log^3(n/\epsilon))$. In many practical federated applications, the nodes are power limited, and hence their computation cost, and communication cost, needs to be low. In terms of total algorithm speed, communication cost/time is often the main bottleneck. The computational cost at the center is a lesser concern.

3) Comparison With Standard Federated Power Method in the No-Attack Setting: Observe that, for a given normalized singular value gap, the sample complexity (lower bound on q) needed by the above result is order nrL/ϵ^2 while that needed for standard PCA (without Byzantine nodes) is order n/ϵ^2 [56, Remark 4.7.2]. The reason we need an extra factor of L is because we are computing the individual node estimates using $\widetilde{q} = q/L$ data points and we need each of the node estimates to be accurate (to ensure that their "median" is accurate). This extra factor of L is needed also in other work that uses (geometric) median, e.g., [16] needs this too. The reason we need an extra factor of r is because we need use Frobenius subspace distance, $\boldsymbol{SD}_F$, to develop and analyze the geometric median step of Subspace Median. The bound provided by the Davis-Kahan sin Theta theorem for $\boldsymbol{SD}_F$ needs an extra factor of $\sqrt{r}$.

Algorithm 3 Subspace Median-of-Means. Recall that $_{(\vartheta,\ell)} =_{(\vartheta-1)\rho+\ell}$.

Input: Batch $\boldsymbol{D}_{(\vartheta)} = [\boldsymbol{D}_{(\vartheta,1)}, \boldsymbol{D}_{(\vartheta,2)}, \ldots, \boldsymbol{D}_{(\vartheta,\rho)}]$, $\vartheta \in [\tilde{L}]$.
Parameters: T_{pow}
Central Server
Randomly initialize $\boldsymbol{U}_{rand}$ with i.i.d standard Gaussian entries. Set $\boldsymbol{U}_{(\vartheta)} = \boldsymbol{U}_{rand}$.
for $t \in [T_{pow}]$ **do**
  Nodes $\ell = 1, \ldots, L$
  Compute $\tilde{\boldsymbol{U}}_{(\vartheta,\ell)} \leftarrow \boldsymbol{D}_{(\vartheta,\ell)}\boldsymbol{D}_{(\vartheta,\ell)}^\top \boldsymbol{U}_{(\vartheta)}$, $\ell \in (\vartheta-1)\rho + [\rho]$, $\vartheta \in [\tilde{L}]$. Push $\tilde{\boldsymbol{U}}_{(\vartheta,\ell)}$ to center.
  Central Server
  Compute $\boldsymbol{U}_{(\vartheta)} \leftarrow QR(\sum_{\ell=1}^{\rho} \tilde{\boldsymbol{U}}_{(\vartheta,\ell)})$, $\vartheta \in [\tilde{L}]$
  Push $\boldsymbol{U}_{(\vartheta)}$ to nodes $\ell \in (\vartheta-1)\rho + [\rho]$.
end for
Use Algorithm 1 for the input $\{\boldsymbol{U}_{(\vartheta)}\}_{\vartheta=1}^{\tilde{L}}$
Output $\boldsymbol{U}_{out}$.

The per-node computational cost of standard federated PCA is $n\widetilde{q}rT_{pow}$ while that for SubsMed is $n\widetilde{q}rT_{pow} + n^2LT_{GM}$. Ignoring log factors and treating the singular value gap as a numerical constant (ignoring T_{pow} and T_{GM}), letting $\epsilon = c$, and substituting the respective lower bounds on $\widetilde{q}$, the PCA cost is n^2r while that for SubsMed for Byzantine-resilient PCA is $\max(n^2r^2, n^2L) = n^2\max(r^2, L)$. Thus the computational cost is only $\max(r, L/r)$ times higher.

We summarize the comparisons in Table I.

V. APPLICATION 2: HORIZONTALLY FEDERATED LRCS

A. Problem Setting

1) Basic Problem: The LRCS problem involves recovering an $n \times q$ rank-r matrix $\boldsymbol{X}^* = [\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \ldots, \boldsymbol{x}_q^*]$, with $r \ll \min(q,n)$, from $\boldsymbol{y}_k := \boldsymbol{A}_k\boldsymbol{x}_k^*$, $k \in [q]$ when $\boldsymbol{y}_k$ is an m-length

vector with $m < n$, and the measurement matrices $\boldsymbol{A}_k$ are known and independent and identically distributed (i.i.d.) over k. We assume that each $\boldsymbol{A}_k$ is a "random Gaussian" matrix, i.e., each entry of it is i.i.d. standard Gaussian. Let $\boldsymbol{X}^* \overset{SVD}{=} \boldsymbol{U}^*\boldsymbol{\Sigma}^*\boldsymbol{V}^{*\top} := \boldsymbol{U}^*\boldsymbol{B}^*$ denote its reduced (rank r) SVD, and $\kappa := \sigma_1^*/\sigma_r^*$ the condition number of $\boldsymbol{\Sigma}^*$. Notice that each measurement $\boldsymbol{y}_{ki}$ is a global function of column $\boldsymbol{x}_k^*$, but not of the entire matrix. As explained in [49], to make it well-posed (allow for correct interpolation across columns), we need the following incoherence assumption on the right singular vectors.

Assumption 2 (Right Singular Vectors' Incoherence): We assume that $\max_k \|\boldsymbol{b}_k^*\| \le \mu\sqrt{r/q}\sigma_1^*$ for a constant $\mu \ge 1$.

2) Horizontal Federation: Consider the $m \times q$ measurements' matrix,

$$\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_q] = [\boldsymbol{A}_1\boldsymbol{x}_1^*, \boldsymbol{A}_2\boldsymbol{x}_2^*, \dots, \boldsymbol{A}_q\boldsymbol{x}_q^*].$$

We assume that there are a total of L nodes and each node observes a different disjoint subset of $\widetilde{m}$ rows of $\boldsymbol{Y}$. Denote the set of indices of the rows available at node ℓ by $\mathcal{S}_\ell$. Thus $|\mathcal{S}_\ell| = \widetilde{m} = m/L$. We assume that node ℓ has access to $\boldsymbol{Y}_\ell$ and $\{(A_k)_\ell, k \in [q]\}$. Here $(\boldsymbol{A}_k)_\ell := \boldsymbol{I}_{\mathcal{S}_\ell}^\top \boldsymbol{A}_k$ is $\widetilde{m} \times n$ and $\boldsymbol{Y}_\ell := \boldsymbol{I}_{\mathcal{S}_\ell}^\top \boldsymbol{Y}$ is of size $\widetilde{m} \times q$ with $\widetilde{m} = m/L$, and with $(\boldsymbol{y}_k)_\ell := (\boldsymbol{A}_k)_\ell \boldsymbol{x}_k^*$ for all $k \in [q]$

Observe that the sub-matrices of rows of $\boldsymbol{Y}$, $\boldsymbol{Y}_\ell$, are identically distributed, in addition to being independent. Consequently, the same is true for the partial gradients computed at the different nodes. This is why, without extra assumptions, we can make our solution Byzantine resilient. On the other hand, column sub-matrices of $\boldsymbol{Y}$ are not identically distributed. In order to obtain provable guarantees for vertical LRCS, we will need extra assumptions that bound on the amount of heterogeneity in the data (and hence in the nodes' partial gradients). This is being studied in ongoing work.

3) Byzantine Nodes: Assumption 1 holds. Also, the set of Byzantine nodes may change at each AltGDmin algorithm iteration.

B. Review of Basic altGDmin [49]

We first explain the basic idea [49] in the simpler no-attack setting. AltGDmin imposes the LR constraint by expressing the unknown matrix $\boldsymbol{X}$ as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{B}$ where $\boldsymbol{U}$ is an $n \times r$ matrix and $\boldsymbol{B}$ is an $r \times q$ matrix. In the absence of attacks, the goal is to minimize

$$f(\boldsymbol{U}, \boldsymbol{B}) := \sum_{k=1}^{q} \|\boldsymbol{y}_k - \boldsymbol{U}\boldsymbol{b}_k\|^2$$

AltGDmin proceeds as follows:

1) *Truncated spectral initialization:* Initialize $\boldsymbol{U}$ (explained below).
2) At each iteration, update $\boldsymbol{B}$ and $\boldsymbol{U}$ as follows:
   a) *Minimization for $\boldsymbol{B}$:* keeping $\boldsymbol{U}$ fixed, update $\boldsymbol{B}$ by solving $\min_{\boldsymbol{B}} f(\boldsymbol{U}, \boldsymbol{B})$. Due to the form of the LRCS measurement model, this minimization decouples across columns, making it a cheap least squares problem of recovering q different r length vectors. It is solved as $\boldsymbol{b}_k \leftarrow (\boldsymbol{A}_k\boldsymbol{U})^\dagger \boldsymbol{y}_k$ for each $k \in [q]$.
   b) *GD for $\boldsymbol{U}$:* keeping $\boldsymbol{B}$ fixed, update $\boldsymbol{U}$ by a GD step, followed by orthonormalizing its columns: $\boldsymbol{U}^+ \leftarrow QR(\boldsymbol{U} - \eta\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}))$. Here $\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}) = \sum_{k\in[q]} \boldsymbol{A}_k^\top(\boldsymbol{A}_k\boldsymbol{U}\boldsymbol{b}_k - \boldsymbol{y}_k)\boldsymbol{b}_k^\top$

The use of full minimization to update $\boldsymbol{B}$ is what helps ensure that AltGDmin provably converges, and that we can show exponential error decay with a constant step size (this statement treats κ as a numerical constant) [49], [50]. Due to the decoupling in this step, its time complexity is only as much as that of computing one gradient w.r.t. $\boldsymbol{U}$. Both steps need time of order $mqnr$. In a federated setting, AltGDmin is also communication-efficient because each node needs to only send nr scalars (gradient w.r.t $\boldsymbol{U}$) at each iteration.

We initialize $\boldsymbol{U}$ by computing the top r singular vectors of

$$\boldsymbol{X}_0 := \sum_k \boldsymbol{A}_k^\top(\boldsymbol{y}_k)_{\mathrm{trunc}}(\alpha)\boldsymbol{e}_k^\top, \text{ where } \boldsymbol{y}_{\mathrm{trunc}}(\alpha) := (\boldsymbol{y} \circ \mathbb{1}_{|\boldsymbol{y}| \le \sqrt{\alpha}})$$

Here $\alpha := 9\kappa^2\mu^2 \sum_k \|\boldsymbol{y}_k\|^2/mq$ and $\boldsymbol{y}_{\mathrm{trunc}}(\alpha)$ is a truncated version of the vector $\boldsymbol{y}$ obtained by zeroing out entries of $\boldsymbol{y}$ with magnitude larger than α (the notation $|\boldsymbol{y}|$ means $|\boldsymbol{y}|_i = |y_i|$ for each entry i, the notation $\mathbb{1}_{\boldsymbol{z}\le\alpha}$ returns a 1-0 vector with 1 where $\boldsymbol{z}_j < \alpha$ and zero everywhere else, and $\boldsymbol{z}_1 \circ \boldsymbol{z}_2$ is the Hadamard product between the two vectors, i.e., the ".*" operation in MATLAB)

Sample-splitting is assumed to prove the guarantees. This means the following: we use a different independent set of measurements and measurement matrices $\boldsymbol{y}_k, \boldsymbol{A}_k, k \in [q]$ for each new update of $\boldsymbol{U}$ and of $\boldsymbol{B}$. We also use a different independent set for computing the initialization threshold α.

All expected values used below are expectations conditioned on past estimates (which are functions of past measurement matrices and measurements, $\boldsymbol{A}_k, \boldsymbol{y}_k$). For example, $\mathbb{E}[(\nabla_U f)_\ell]$ conditions on the values of $\boldsymbol{U}, \boldsymbol{B}_\ell$ used to compute it. This is also the reason why $\mathbb{E}[(\nabla_U f)_\ell]$ is different for different nodes; see Lemma 2.

C. Resilient Federated Spectral Initialization

This consists of two steps. First the truncation threshold $\alpha = \frac{\tilde{C}}{mq}\sum_k \sum_i y_{ki}^2$ which is a scalar needs to be computed. This is simple: each node computes $\alpha_\ell = \frac{\tilde{C}}{mq}\sum_k \sum_{i\in\mathcal{S}_\ell} (y_\ell)_{ki}^2$ and sends it to the center which computes their median.

Next, we need to compute $\boldsymbol{U}_0$ which is the matrix of top r left singular vectors of $\boldsymbol{X}_0$, and hence also of $\boldsymbol{X}_0\boldsymbol{X}_0^\top$. Node ℓ has data to compute the $n \times q$ matrix $(\boldsymbol{X}_0)_\ell$, defined as

$$(\boldsymbol{X}_0)_\ell := \sum_{k=1}^{q} (\boldsymbol{A}_k)_\ell{}^\top ((\boldsymbol{y}_k)_\ell)_{\mathrm{trunc}} \boldsymbol{e}_k^\top, \tag{2}$$

Observe that $\boldsymbol{X}_0 = \sum_\ell (\boldsymbol{X}_0)_\ell$. If all nodes were good (non-Byzantine), we would use this fact to implement the federated power method for this case. However, some nodes can be Byzantine and hence this approach will not work. For reasons similar to those explained in Sec. III, (i) an obvious GM-based modification of the federated power method will not work either, and (ii) nodes cannot send the entire $(\boldsymbol{X}_0)_\ell$ (this is too expensive to communicate). We instead use Subspace

Algorithm 4 Byz-AltGDmin: Initialization Using Subspace Median

Input $\boldsymbol{Y}_\ell$ and $(\boldsymbol{A}_k)_\ell$
Parameters T_{GM}, T_{pow}
Nodes $\ell = 1, \dots, L$
Compute $\alpha_\ell \leftarrow \frac{9\kappa^2\mu^2}{\widetilde{m}q} \sum_k \|(\boldsymbol{y}_k)_\ell\|^2$.
Central Server
$\alpha \leftarrow \text{Median}\{\alpha_\ell\}_{\ell=1}^L$
Nodes $\ell = 1, \dots, L$
Compute $(\boldsymbol{U}_0)_\ell \leftarrow$ top-r-singular vectors of $(\boldsymbol{X}_0)_\ell$ defined in (2) (use power method with T_{pow} iterations).
Central Server
Use Algorithm 1 (Subspace-Median) with parameter T_{GM} on $(\boldsymbol{U}_0)_\ell$, $\ell \in [L]$.
Output $\boldsymbol{U}_{out}$.

Median, Algorithm 1, applied to $\boldsymbol{D}_\ell = (\boldsymbol{X}_0)_\ell$. This is both communication-efficient and sample-efficient. It can be shown that it will work under a sample complexity lower bound that is comparable to that needed in the attack-free setting. We summarize this in Algorithm 4. We can obtain a guarantee for this approach by applying Theorem 3.1 with $\boldsymbol{\Phi}_\ell \equiv (\boldsymbol{X}_0)_\ell (\boldsymbol{X}_0)_\ell^\top / q$ and using the results from [49] and [50] to ensure that the assumption needed by Theorem 3.1 holds. We directly state a guarantee for the GM of means estimator developed next. The guarantee for Algorithm 4 is a special case of that for the GM of means estimator developed next with $\tilde{L} = L$, and thus its guarantee is also given by Corollary 3 with $\tilde{L} = L$.

D. Resilient Federated Spectral Initialization: Horizontal Subspace-MoM

As explained earlier for PCA, the use of just (geometric) median wastes samples. Hence, we develop a median-of-means estimator. For a parameter $\tilde{L} \leq L$, we would like to form $\tilde{L}$ mini-batches of $\rho = L/\tilde{L}$ nodes; w.l.o.g. ρ is an integer. In our current setting, the data is horizontally federated. This requires a different approach to combine samples than what we used for PCA in Sec. IV-B. Here, each node can compute the $n \times q$ matrix $(\boldsymbol{X}_0)_\ell$. Combining samples means combining the rows of $(\boldsymbol{A}_k)_\ell$ and $(\boldsymbol{y}_k)_\ell$ for ρ nodes to obtain $(\boldsymbol{X}_0)_{(\vartheta)}$ with k-th column given by $\sum_{\ell=1}^{\rho} (\boldsymbol{A}_k)_{(\vartheta,\ell)}^\top (\boldsymbol{y}_{k,\text{trunc}})_{(\vartheta,\ell)}/\rho$. Recall that $(\vartheta, \ell) = (\vartheta - 1)\rho + \ell)$. To compute this in a communication-efficient and private fashion, we use a horizontally federated power method for each of the $\tilde{L}$ mini-batches. The output of each of these power methods is $\boldsymbol{U}_{(\vartheta)}$, $\vartheta \in [\tilde{L}]$. These are then input to the subspace-median algorithm, Algorithm 1 to obtain the final subspace estimate $\boldsymbol{U}_{out}$. To explain the federation details simply, we explain them for $\vartheta = 1$. The power method needs to federate $\boldsymbol{U} \leftarrow QR((\boldsymbol{X}_0)_{(1)}(\boldsymbol{X}_0)_{(1)}^\top \boldsymbol{U}) = QR(\sum_{\ell'=1}^{\rho} (\boldsymbol{X}_0)_{\ell'} (\sum_{\ell=1}^{\rho} (\boldsymbol{X}_0)_\ell^\top \boldsymbol{U}))$. This needs two steps of information exchange between the nodes and center at each power method iteration. In the first step, we compute $\boldsymbol{V} = \sum_{\ell \in [\rho]} (\boldsymbol{X}_0)_\ell{}^\top \boldsymbol{U}$, and in the second one we compute $\tilde{\boldsymbol{U}} = \sum_{\ell \in [\rho]} (\boldsymbol{X}_0)_\ell \boldsymbol{V}$, followed by its QR decomposition.

We summarize the complete algorithm in Algorithm 5. As long as the same set of τL nodes are Byzantine for all the power method iterations needed for the initialization step, we can prove the following result for it[3]. This follows as a corollary of Theorem 3.1 and the lemmas proved in [49] and [50] for the attack-free case.

Corollary 3 (Initialization Using Subspace-GMoM): Consider the Initialization steps (lines 3-22) of Algorithm 5 with $T_{GM} = C \log(\frac{Lr}{\delta_0})$ and $T_{pow} = C\kappa^2 \log(\frac{n}{\delta_0})$. Assume that Assumption 2 hold. Assume also that the set of Byzantine nodes remains fixed for all iterations in this algorithm and the size of this set is at most τL with $\tau < 0.4\tilde{L}/L$. Pick a $\delta_0 < 1$ and an $\tilde{L} < L$ such that L is a multiple of $\tilde{L}$. If

$$mq \geq C\tilde{L} \cdot \kappa^6 \mu^2 (n+q) r^2 / \delta_0^2,$$

then w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4 - \tau, \exp(-c(n+q)) + n^{-10})) - L\exp(-\tilde{c}\widetilde{m}q\delta_0^2/r^2\kappa^4)$,

$$\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_{out}) \leq \delta_0.$$

The communication cost per node is order $nr \cdot T_{pow} = \kappa^2 nr \log(\frac{n}{\delta_0})$.

Proof: This follows by applying Theorem 3.1 on $\boldsymbol{\Phi}_{(\vartheta)} = \sum_{\ell=1}^{\rho} (\boldsymbol{X}_0)_{(\vartheta,\ell)} (\boldsymbol{X}_0)_{(\vartheta,\ell)}^\top / \rho$ and $\boldsymbol{\Phi}^* = \mathbb{E}[(\boldsymbol{X}_0)_\ell | \alpha] \mathbb{E}[(\boldsymbol{X}_0)_\ell | \alpha]^\top$ for $\vartheta \in [\tilde{L}]$ and using the results from [49] and [50] to ensure that the assumption needed by Theorem 3.1 holds.

The idea is almost exactly the same as for the special case $\tilde{L} = L$. This case is simpler notation-wise and hence we provide a proof for this case in Appendix A-B. The main idea is as follows. Let $\boldsymbol{D}(\alpha)$ be the positive entries' diagonal matrix defined in [49, Lemma 3.8]. We use [49, Lemma 3.8] and [49, Fact 3.9] to show that $\mathbb{E}[(\boldsymbol{X}_0)_\ell | \alpha] = \boldsymbol{X}^* \boldsymbol{D}(\alpha)$ and to bound $\|(\boldsymbol{X}_0)_\ell - \mathbb{E}[(\boldsymbol{X}_0)_\ell | \alpha]\|$. We then use this bound to then get a bound $\|\boldsymbol{\Phi}_\ell - \boldsymbol{\Phi}^*\|$. In the last step, we use an easy median-based modification of [49, Fact 3.7] to remove the conditioning on α. □

E. Byzantine-Resilient Federated AltGDmin: GDmin Iterations

We can make the altGDmin iterations resilient as follows. In the minimization step, each node computes its own estimate $(\boldsymbol{b}_k)_\ell$ of $\boldsymbol{b}_k^*$ as follows:

$$(\boldsymbol{b}_k)_\ell = ((\boldsymbol{A}_k)_\ell \boldsymbol{U})^\dagger (\boldsymbol{y}_k)_\ell, \ k \in [q]$$

Here, $\boldsymbol{M}^\dagger := (\boldsymbol{M}^\top \boldsymbol{M})^{-1} \boldsymbol{M}^\top$. Each node then uses this to compute its estimate of the gradient w.r.t. $\boldsymbol{U}$ as $\nabla f_\ell = \sum_{k \in [q]} (\boldsymbol{A}_k)_\ell^\top ((\boldsymbol{A}_k)_\ell \boldsymbol{U} (\boldsymbol{b}_k)_\ell - (\boldsymbol{y}_k)_\ell)(\boldsymbol{b}_k)_\ell^\top$. The center receives the gradients from the different nodes, computes their GM and uses this for the projected GD step. Since the gradient norms are not bounded, the GM computation needs to be preceded by the thresholding step explained in Sec. VI-A.2.

As before, to improve sample complexity (while reducing Byzantine tolerance), we can replace GM of the gradients by their GM of means: form $\tilde{L}$ batches of size $\rho = L/\tilde{L}$

[3]This assumption can be relaxed if we instead assume that the size of the set of nodes that are Byzantine in any one initialization iteration is at most τL.

each, compute the mean gradient within each batch, compute the GM of the $\tilde{L}$ mean gradients. Use appropriate scaling. We summarize the GMoM algorithm in Algorithm 5. The GM case corresponds to $\tilde{L} = L$. Given a good enough initialization, a small enough fraction of Byzantine nodes, enough samples $\widetilde{m}q$ at each node at each iteration, and assuming that Assumption 2 holds, we can prove the following for the GD iterations.

Theorem 5.3 (AltGDmin-GMoM: Error Decay): Consider the AltGDmin steps of Algorithm 5 with sample-splitting, and with a step-size $\eta \leq 0.5/\sigma_1^{*2}$. Set $T_{GM} = C \log \frac{Lr}{\epsilon}$, $\omega_{GM} = C\widetilde{m}\sigma_r^{*2}$. Assume that Assumptions 1 and 2 holds. If, at each iteration t,

$$mq \geq C_1 \tilde{L}\kappa^4 \mu^2 (n+r) r^2,$$

$\widetilde{m} > C_2 \max(\log q, \log n)$; if $\tau < 0.4\tilde{L}/L$; and if the initial estimate $\boldsymbol{U}_0$ satisfies $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_0) \leq \delta_0 = 0.1/\kappa^2$, then w.p. at least $1 - c_{approxGM} - t\left[Ln^{-10} + \exp(-L\psi(0.4-\tau, n^{-10}))\right]$,

$$\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_{t+1}) \leq \delta_{t+1} := \left(1 - (\eta\sigma_1^{*2})\frac{0.12}{\kappa^2}\right)^{t+1} \delta_0$$

and $\|\boldsymbol{x}_k^* - (\boldsymbol{x}_k)_{t+1}\| \leq \delta_{t+1}\|\boldsymbol{x}_k^*\|$ for all $k \in [q]$.

The communication cost per node is order $nr \cdot T = \kappa^2 nr \log(\frac{1}{\epsilon})$.

Proof: Consider the $\tilde{L} = L$ (GM) special case since this is notationally simpler. The extension for the general $\tilde{L} < L$ (GM of means) case is straightforward. The proof uses the overall approach developed in [50] with the following changes. Let $\ell_1 := (\mathcal{J}_{good})_1$ be *a* non-Byzantine node. We now also need a bound on the Frobenius norm of

$$\text{Err} := \nabla f_{GM} - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})], \quad \ell_1 := (\mathcal{J}_{good})_1$$

that is of the form $c\delta_t \widetilde{m}\sigma_1^{*2}$ for a $c < 1$ w.h.p., under the assumed sample complexity bound. This type of a bound, along with assuming $\delta_0 < c/\sqrt{r}\kappa^2$, helps ensure that the algebra needed for showing exponential decay of the subspace estimation error goes through. We can get the above bound on $\|\text{Err}\|_F$ using Lemma 6 if we can get a similar bound on

$$\max_{\ell \in \mathcal{J}_{good}} \|\nabla f_\ell - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})]\|_F$$

This is proved in the lemma given next.

Lemma 2: Assume $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}) \leq \delta_t < \delta_0$. Then, w.p. at least $1 - \exp\left((n+r) - c\epsilon_1^2 \frac{\widetilde{m}q}{r^2\mu^2}\right) - 2\exp(\log q + r - c\epsilon_1^2 \widetilde{m})$, for all $\ell \in \mathcal{J}_{good}$,

$$\|\nabla f_\ell(\boldsymbol{U}, \boldsymbol{B}_\ell) - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})]\|_F \leq 12.5\epsilon_1 \delta_t \widetilde{m}\sigma_1^{*2}$$

We prove this lemma by noting that

$$\begin{aligned}
&\nabla f_\ell(\boldsymbol{U}, \boldsymbol{B}_\ell) - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})] \\
&= (\nabla f_\ell(\boldsymbol{U}, \boldsymbol{B}_\ell) - \mathbb{E}[\nabla f_\ell(\boldsymbol{U}, \boldsymbol{B}_\ell)]) \\
&+ (\mathbb{E}[\nabla f_\ell(\boldsymbol{U}, \boldsymbol{B}_\ell)] - E[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})])
\end{aligned}$$

The first term can be bounded using standard concentration bounds. The second one requires carefully bounding $\|\boldsymbol{B}_\ell - \boldsymbol{B}_{\ell_1}\|_F$ by using the fact that both $\boldsymbol{B}_\ell, \boldsymbol{B}_{\ell_1}$ are close to $\boldsymbol{G} := \boldsymbol{U}^\top \boldsymbol{X}^*$.

We provide its proof and the complete proof of our Theorem in Appendix B. □

F. Complete Byz-AltGDmin Algorithm

Combining Corollary 3 and Theorem 5.3, and setting $\eta = 0.5/\sigma_1^{*2}$ and $\delta_0 = 0.1/\kappa^2$, we can show that, at iteration $t+1$, $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_{t+1}) \leq \delta_{t+1} = (1 - 0.06/\kappa^2)^{t+1} 0.1/\kappa^2$ whp. Thus, in order for this to be $\leq \epsilon$, we need to set $T = C\kappa^2 \log(1/\epsilon)$. Also, since we are using fresh samples at each iteration (sample-splitting), this also means that our sample complexity needs to be multiplied by T.

We thus have the following final result.

Corollary 4: Consider the complete Algorithm 5 with sample-splitting. Set $T_{pow} = C\log(n\kappa)$, $\eta = 0.5/\sigma_1^{*2}$, $T = C\kappa^2 \log(1/\epsilon)$. Assume that Assumption 2 holds. If the total number of samples per column m, satisfies

$$mq \geq C\tilde{L}\kappa^4 \mu^2 (n+q) r^2 \log(1/\epsilon)$$

and $m > C\kappa^2 \max(\log q, \log n) \log(1/\epsilon)$; if at most τL nodes are Byzantine with $\tau < 0.4\tilde{L}/L$, if the set of Byzantine nodes remains fixed for the initialization step power method (but can vary for the GDmin iterations); then, w.p. at least $1 - TLn^{-10}$, $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}_T) \leq \epsilon$, and $\|\boldsymbol{x}_k - \boldsymbol{x}_k^*\| \leq \epsilon\|\boldsymbol{x}_k^*\|$ for all $k \in [q]$.

The communication cost per node is order $\kappa^2 nr \log(\frac{n}{\epsilon})$. The computational cost at any node is order $\kappa^2 \widetilde{m} nqr \log(\frac{n}{\epsilon})$ while that at the center is $n^2 \tilde{L} \log^3(\tilde{L}r/\epsilon)$.

The above result shows that, under exactly one assumption (Assumption 2), if each node has enough samples $\widetilde{m}$ ($\widetilde{m}$ is of order $(n+q)r^2(\tilde{L}/L)$ times log factors); if the number of Byzantine nodes is less than $(0.4\tilde{L}/L)$ times the total number of nodes, then our algorithm can recover each column of the LR matrix $\boldsymbol{X}^*$ to ϵ accuracy whp. To our best knowledge, the above is the first guarantee for Byzantine resiliency for any type of low rank matrix recovery problems studied in a federated setting.

Observe that the above result needs total sample complexity that is only $\tilde{L}$ times that for basic AltGDmin [49].

VI. Proofs for Sec III and IV

All the proofs given below rely on the lemma for using GM for robust estimation borrowed from [16]. We give these lemmas in the section below, followed by two corollaries that will be used in our proofs.

A. Using GM for Robust Estimation

The goal of robust estimation is to get a reliable estimate of a vector quantity $\tilde{\boldsymbol{z}}$ using L individual estimates of it, denoted $\boldsymbol{z}_\ell$, when most of the estimates are good, but a few can be arbitrarily corrupted or modified by Byzantine attackers. A good approach to do this is to use the GM. The following lemma, which is a minor modification of [16, Lemma 2.1], studies this[4]

Lemma 3: Consider $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_\ell, \ldots \boldsymbol{z}_L\}$ with $\boldsymbol{z}_\ell \subseteq \Re^n$. Let $\boldsymbol{z}_{GM}^*$ denote their GM and let $\boldsymbol{z}_{GM}$ denote their $(1 + \epsilon_{GM})$ approximate GM estimate computed using [60, Algorithm 1]. Fix an $\alpha \in (0, 1/2)$. Suppose that the following holds for at least $(1-\alpha)L$ $\boldsymbol{z}_\ell$s:

$$\|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \leq \epsilon \|\tilde{\boldsymbol{z}}\|.$$

[4] [16, Lemma 2.1] does not provide an algorithm for approximating the Geometric Median; we combine their result with Claim 1 to provide this.

Algorithm 5 Byz-AltGDmin: Complete GMoM based algorithm

Input: Batch ϑ : $\{(\boldsymbol{A}_k)_\ell, \boldsymbol{Y}_\ell, k \in [q]\}$, $\ell \in [L]$
Parameters: T_{pow}, T_{GM},
Initialization using Subspace MoM:
Nodes $\ell = 1, \dots, L$
Compute $\alpha_\ell \leftarrow \frac{9\kappa^2\mu^2}{\widetilde{m}q} \sum_k \|(\boldsymbol{y}_k)_\ell\|^2$.
Central Server
$\alpha \leftarrow \text{Median}\{\alpha_{(\vartheta)}\}_{\vartheta=1}^{\tilde{L}}$, where $\alpha_{(\vartheta)} = \sum_{\ell=1}^{\rho} \alpha_{(\vartheta,\ell)}/\rho$
Central Server
Let $\boldsymbol{U}_0 = \boldsymbol{U}_{rand}$ where $\boldsymbol{U}_{rand}$ is an $n \times r$ matrix with i.i.d standard Gaussian entries.
for $\tau \in [T_{pow}]$ **do**
  Nodes $\ell = 1, \dots, L$
  Compute $\boldsymbol{V}_\ell \leftarrow (\boldsymbol{X}_0)_\ell^\top (\boldsymbol{U}_{(\vartheta)})_{\tau-1}$ for $\ell \in (\vartheta-1)\rho + [\rho]$, $\vartheta \in [\tilde{L}]$. Push to center.
  Central Server
  Compute $\boldsymbol{V}_{(\vartheta)} \leftarrow \sum_{\ell=1}^{\rho} \boldsymbol{V}_{(\vartheta-1)\rho+\ell}$
  Push $\boldsymbol{V}_{(\vartheta)}$ to nodes $\ell \in (\vartheta-1)\rho + [\rho]$.
  Nodes $\ell = 1, \dots, L$
  Compute $\boldsymbol{U}_\ell \leftarrow \sum_k (\boldsymbol{X}_0)_\ell \boldsymbol{V}_{(\vartheta)}$ for $\ell \in (\vartheta-1)\rho + [\rho]$, $\vartheta \in [\tilde{L}]$. Push to center.
  Central Server
  Compute $\boldsymbol{U}_{(\vartheta)} \leftarrow QR(\sum_{\ell=1}^{\rho} \boldsymbol{U}_{(\vartheta-1)\rho+\ell})$
  Push $\boldsymbol{U}_{(\vartheta)}$ to nodes $\ell \in (\vartheta-1)\rho + [\rho]$.
end for
Apply Algorithm 1 on $\{U_\vartheta\}_{\vartheta=1}^{\tilde{L}}$ to get $\boldsymbol{U}_{out}$.
Set $\boldsymbol{U}_0 \leftarrow \boldsymbol{U}_{out}$
AltGDmin Iterations:
for $t = 1$ to T **do**
  Nodes $\ell = 1, \dots, L$
  Set $\boldsymbol{U} \leftarrow \boldsymbol{U}_{t-1}$
  $(\boldsymbol{b}_k)_\ell \leftarrow ((\boldsymbol{A}_k)_\ell \boldsymbol{U})^\dagger (\boldsymbol{y}_k)_\ell, \ \forall\ k \in [q]$
  $(\boldsymbol{x}_k)_\ell \leftarrow \boldsymbol{U}(\boldsymbol{b}_k)_\ell, \ \forall\ k \in [q]$
  $(\nabla f)_\ell \leftarrow \sum_{k\in[q]} (\boldsymbol{A}_k)_\ell^\top ((\boldsymbol{A}_k)_\ell \boldsymbol{U}(\boldsymbol{b}_k)_\ell - (\boldsymbol{y}_k)_\ell)(\boldsymbol{b}_k)_\ell{}^\top$, $\forall\ k \in [q]$
  Push ∇f_ℓ to center
  Central Server
  Compute $\nabla f_{(\vartheta)} \leftarrow \sum_{\ell\in\vartheta} \nabla f_\ell$
  $\nabla f_{GM} \leftarrow approxGM\left(\{vec(\nabla f_{(\vartheta)}),\ \vartheta \in [\tilde{L}]\} \backslash \{\vartheta : \|\nabla f_{(\vartheta)}\|_F > \omega_{GM}\}\right)$
  (Use [60, Algorithm 1] with T_{GM} iterations on the set of $\nabla f_{(\vartheta)}$s whose Frobenius norm is below ω_{GM})
  Compute $\boldsymbol{U}^+ \leftarrow QR(\boldsymbol{U}_{t-1} - \frac{\eta}{\rho\widetilde{m}} \nabla f_{GM})$
  return Set $\boldsymbol{U}_t \leftarrow \boldsymbol{U}^+$. Push $\boldsymbol{U}_t$ to nodes.
end for
Output $\boldsymbol{U}_T$.

Let $C_\alpha := \frac{2(1-\alpha)}{1-2\alpha}$. Then, w.p. at least $1 - c_{approxGM}$,

$$\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| \le C_\alpha \epsilon \|\tilde{\boldsymbol{z}}\| + \epsilon_{GM} \frac{\sum_{\ell=1}^{L} \|\boldsymbol{z}_{GM}^* - \boldsymbol{z}_\ell\|}{(1-2\alpha)L}$$
$$\le C_\alpha \epsilon \|\tilde{\boldsymbol{z}}\| + \epsilon_{GM} \frac{\max_{\ell\in[L]} \|\boldsymbol{z}_\ell\|}{1-2\alpha}$$

The number of iterations needed for computing $\boldsymbol{z}_{GM}$ is $T_{GM} = C \log(\frac{L}{\epsilon_{GM}})$, and the time complexity is $O\left(nL \log^3(\frac{L}{\epsilon_{GM}})\right)$.

The second inequality follows because, using the exact GM definition, $\sum_\ell \|\boldsymbol{z}_{GM}^* - \boldsymbol{z}_\ell\| \le \sum_\ell \|\boldsymbol{0} - \boldsymbol{z}_\ell\| = \sum_\ell \|\boldsymbol{z}_\ell\|$ and $\sum_\ell \|\boldsymbol{z}_\ell\| \le L \max_\ell \|\boldsymbol{z}_\ell\|$. To understand this lemma simply, fix the value α to 0.4. Then $C_\alpha = 6$. We can also fix $\epsilon_{GM} = \epsilon$. Then, it says the following. If at least 60% of the L estimates are ϵ close to $\tilde{\boldsymbol{z}}$, then, the $(1+\epsilon)$ approximate GM, $\boldsymbol{z}_{GM}$, is $11\epsilon \max(\|\tilde{\boldsymbol{z}}\|, \max_{\ell\in[L]} \|\boldsymbol{z}_\ell\|)$ close to $\tilde{\boldsymbol{z}}$. The next lemma follows using the above lemma and is a minor modification of [16, Lemma 3.5]. It fixes $\alpha = 0.4$ and considers the case when most estimates are good with high probability (w.h.p.). We provide a short proof of it in Appendix D-D.

Lemma 4: Let $\boldsymbol{z}_\ell \subseteq \Re^n$, for $\ell \in [L]$ and let $\boldsymbol{z}_{GM}$ denote a $(1+\epsilon_{GM})$ approximate GM computed using [60, Algorithm 1]. For a $\tau < 0.4$, suppose that, for at least $(1-\tau)L$ $\boldsymbol{z}_\ell$'s,

$$\Pr\{\|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \le \epsilon \|\tilde{\boldsymbol{z}}\|\} \ge 1 - p$$

Then, w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4-\tau, p))$,

$$\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| \le 6\epsilon \|\tilde{\boldsymbol{z}}\| + 5\epsilon_{GM} \max_{\ell\in[L]} \|\boldsymbol{z}_\ell\|$$

where $\psi(a,b) = (1-a)\log\frac{1-a}{1-b} + a\log\frac{a}{b}$. The number of iterations needed for computing $\boldsymbol{z}_{GM}$ is $T_{GM} = C\log(\frac{L}{\epsilon_{GM}})$, and the time complexity is $O\left(nL\log^3(\frac{L}{\epsilon_{GM}})\right)$.

Suppose that, for a $\tau < 0.4$, at least $(1-\tau)L$ $\boldsymbol{z}_\ell$s are "good" (are ϵ close to $\tilde{\boldsymbol{z}}$) whp. Let $\epsilon_{GM} = \epsilon$ and suppose that all $\boldsymbol{z}_\ell$'s, including the corrupted ones, are bounded in 2-norm by $\|\tilde{\boldsymbol{z}}\|$. Then, the $(1+\epsilon)$-approximate GM is about $11\epsilon\|\tilde{\boldsymbol{z}}\|$ close to $\tilde{\boldsymbol{z}}$ with at least constant probability. If the GM is approximated with probability 1, i.e., if $c_{approxGM} = 0$, then, the above result says that, for p small enough and large L, the reliability of the GM is actually higher than that of the individual good estimates. For example, for a $p < 0.01$, the probability is at least $1 - p^{L(0.4-\tau)}$. The increase depends on $(0.4-\tau)$ and L, e.g., if $\tau \ge 0.2$ and $L \ge 10$, then, the probability is at least $1 - p^{0.2\ L} \ge 1 - p^2$.

1) Corollary for Bounded z_ℓs: In settings where all $\boldsymbol{z}_\ell$'s are bounded, we have the following corollary of Lemma 4.

Corollary 5: In the setting of Lemma 4, if $\max_\ell \|\boldsymbol{z}_\ell\| \le \|\tilde{\boldsymbol{z}}\|$, then $\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| \le 11\epsilon\|\tilde{\boldsymbol{z}}\|$ with above probability. The number of iterations needed is $T_{GM} = C\log(\frac{L}{\epsilon_{GM}})$, and the time complexity is $O\left(nL\log^3(\frac{L}{\epsilon_{GM}})\right)$.
We use this for analyzing the Subspace Median algorithm in which the $\boldsymbol{z}_\ell$s are vectorized projection matrices from the different nodes.

2) Corollary for Unbounded z_ℓs: When some $\boldsymbol{z}_\ell$s may not be bounded, we need an extra thresholding step. Observe that, from the assumption in Lemma 4, w.p. at least $1 - Lp$, the good $\boldsymbol{z}_\ell$s are bounded by $(1+\epsilon)\|\tilde{\boldsymbol{z}}\|$. Thus, to get a set of $\boldsymbol{z}_\ell$'s that are bounded in norm, while not eliminating any of the good ones, we can create a new set that only contains $\boldsymbol{z}_\ell$'s with norm smaller than threshold $\omega_{GM} = (1+\epsilon)\|\tilde{\boldsymbol{z}}\|$. In other words, we compute the GM of the set $\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_L\} \setminus \{\boldsymbol{z}_\ell : \|\boldsymbol{z}_\ell\| > (1+\epsilon)\|\tilde{\boldsymbol{z}}\|\}$ as the input to the GM computation algorithm [60, Algorithm 1]. More generally, ω can be set to $C\|\tilde{\boldsymbol{z}}\|$ for any $C > 1$. In practice, to set the threshold, we only need to have an estimate of the norm of the unknown quantity $\tilde{\boldsymbol{z}}$ that we are trying to estimate.

We have the following corollary of Lemma 6 for this setting.

Corollary 6: Let $\boldsymbol{z}_{GM}$ denote a $(1+\epsilon_{GM})$ approximate GM of $\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_L\} \setminus \{\boldsymbol{z}_\ell : \|\boldsymbol{z}_\ell\| > \omega_{GM}\}$, all vectors are in $\Re^n$. Set $\omega_{GM} = (1+\epsilon)\|\tilde{\boldsymbol{z}}\|$. For a $\tau < 0.4$, suppose that, for at least $(1-\tau)L$ $\boldsymbol{z}_\ell$'s,

$$\Pr\{\|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \le \epsilon\|\tilde{\boldsymbol{z}}\|\} \ge 1-p$$

Then, w.p. at least $1 - c_{approxGM} - Lp - \exp(-L\psi(0.4-\tau, p))$,

$$\begin{aligned}\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| &\le 6\epsilon\|\tilde{\boldsymbol{z}}\| + 5\epsilon_{GM}(1+\epsilon)\|\tilde{\boldsymbol{z}}\| \\ &< 14\max(\epsilon, \epsilon_{GM})\|\tilde{\boldsymbol{z}}\|\end{aligned}$$

The number of iterations needed is $T_{GM} = C\log(\frac{L}{\epsilon_{GM}})$, and the time complexity is $O\left(nL\log^3(\frac{L}{\epsilon_{GM}})\right)$.
We use this for analyzing ResPowMeth with $\boldsymbol{z}_\ell$ being the vectorized $\boldsymbol{\Phi}_\ell \boldsymbol{U}_\tau$. It is also used later for analyzing the GD step of the alternating GD and minimization (altGDmin) algorithm for solving the LRCS problem.

B. Proof of Lemma 1

Since $\boldsymbol{SD}_F(\boldsymbol{U}_\ell, \boldsymbol{U}^*) = (1/\sqrt{2})\|\mathcal{P}_{\boldsymbol{U}_\ell} - \mathcal{P}_{\boldsymbol{U}^*}\|_F$ [61, Lemma 2.5], thus, the lemma assumption implies that $\max_{\ell\in\mathcal{J}_{good}} \|\mathcal{P}_{\boldsymbol{U}_\ell} - \mathcal{P}_{\boldsymbol{U}^*}\|_F \le \sqrt{2}\delta$.

Observe that $\|\mathcal{P}_{\boldsymbol{U}}\|_F \le \sqrt{r}$ for any matrix $\boldsymbol{U}$ with orthonormal columns. Thus $\|\mathcal{P}_{\boldsymbol{U}_\ell}\| \le \sqrt{r}$ for all ℓ including the Byzantine ones (recall that we orthonormalize the received $\hat{\boldsymbol{U}}_\ell$'s using QR at the center before computing $\mathcal{P}_{\boldsymbol{U}_\ell}$). Hence, using GM Lemma 4, we have w.p. at least $1 - c_{approxGM} - \exp\left(-L\psi\left(0.4-\tau, p\right)\right)$

$$\|\mathcal{P}_{GM} - \mathcal{P}_{\boldsymbol{U}^*}\|_F \le 6\sqrt{2}\delta + 5\epsilon_{GM}\sqrt{r} \tag{3}$$

Here $\mathcal{P}_{GM} = GM\{\mathcal{P}_{\boldsymbol{U}_\ell}, \ell \in [L]\}$. Thus,

$$\begin{aligned}&\max_{\ell\in\mathcal{J}_{good}} \|\mathcal{P}_{\boldsymbol{U}_\ell} - \mathcal{P}_{GM}\|_F \\ &\le \max_{\ell\in\mathcal{J}_{good}} \|\mathcal{P}_{\boldsymbol{U}_\ell} - \mathcal{P}_{\boldsymbol{U}^*}\|_F + \|\mathcal{P}_{GM} - \mathcal{P}_{\boldsymbol{U}^*}\|_F \\ &\le \sqrt{2}\delta + 6\sqrt{2}\delta + 5\epsilon_{GM}\sqrt{r} = 7\sqrt{2}\delta + 5\epsilon_{GM}\sqrt{r}\end{aligned}$$

w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4-\tau, p))$.

Next we bound the $\boldsymbol{SD}$ between $\mathcal{P}_{GM}$ and the node closest to it. This is denoted ℓ_{best} in the algorithm.

$$\begin{aligned}\|\mathcal{P}_{\boldsymbol{U}_{\ell_{best}}} - \mathcal{P}_{GM}\|_F &= \min_\ell \|\mathcal{P}_{\boldsymbol{U}_\ell} - \mathcal{P}_{GM}\|_F \\ &\le \min_{\ell\in\mathcal{J}_{good}} \|\mathcal{P}_{\boldsymbol{U}_\ell} - \mathcal{P}_{GM}\|_F \\ &\le \max_{\ell\in\mathcal{J}_{good}} \|\mathcal{P}_{\boldsymbol{U}_\ell} - \mathcal{P}_{GM}\|_F \\ &\le 7\sqrt{2}\delta + 5\epsilon_{GM}\sqrt{r}\end{aligned}$$

In this we used $\mathcal{J}_{good} \subseteq [L]$ and hence the minimum value over all L is smaller than that over all $\ell \in \mathcal{J}_{good}$. We use this to bound the $\boldsymbol{SD}$ between $\boldsymbol{U}_{\ell_{best}}$ and $\boldsymbol{U}^*$.

$$\begin{aligned}&\|\mathcal{P}_{\boldsymbol{U}_{\ell_{best}}} - \mathcal{P}_{\boldsymbol{U}^*}\|_F \\ &\le \|\mathcal{P}_{\boldsymbol{U}_{\ell_{best}}} - GM\|_F + \|GM - \mathcal{P}_{\boldsymbol{U}^*}\|_F \\ &\le 7\sqrt{2}\delta + 5\epsilon_{GM}\sqrt{r} + 6\sqrt{2}\delta + 5\epsilon_{GM}\sqrt{r} \\ &\le 13\sqrt{2}\delta + 10\epsilon_{GM}\sqrt{r}\end{aligned} \tag{4}$$

Set $\epsilon_{GM} = \delta\sqrt{2}/\sqrt{r}$. Thus, we have that, w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4-\tau, p))$,

$$\|\mathcal{P}_{\boldsymbol{U}_{\ell_{best}}} - \mathcal{P}_{\boldsymbol{U}^*}\|_F \le 23\sqrt{2}\delta$$

This then implies that $\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) = \boldsymbol{SD}_F(\boldsymbol{U}_{\ell_{best}}, \boldsymbol{U}^*) \le 23\delta$ since $\boldsymbol{U}_{out} = \boldsymbol{U}_{\ell_{best}}$.

Note: It is possible that ℓ_{best} is not a good node (we cannot prove that it is). This is why the above steps are needed to bound $\|\mathcal{P}_{\boldsymbol{U}_{\ell_{best}}} - \mathcal{P}_{\boldsymbol{U}^*}\|_F$.

C. Proof of Theorem 3.1, Exact SVD at the Nodes

The version of Davis-Kahan $\sin\Theta$ theorem [62] stated next is taken from [61, Corollary 2.8].

Claim 2 (Davis-Kahan $\sin\Theta$ theorem [61], [62]): Let $\boldsymbol{\Phi}^*, \boldsymbol{\Phi}$ be $n \times n$ symmetric matrices with $\boldsymbol{U}^* \in \Re^{n\times r}$, $\boldsymbol{U} \in \Re^{n\times r}$ being the matrices of top r singular/eigen vectors of $\boldsymbol{\Phi}^*, \boldsymbol{\Phi}$ respectively. Let $\sigma_1^* \ge \dots \ge \sigma_n^*$ be the eigenvalues of $\boldsymbol{\Phi}^*$. If $\sigma_r^* - \sigma_{r+1}^* > 0$ and $\|\boldsymbol{\Phi} - \boldsymbol{\Phi}^*\| \le \left(1 - \frac{1}{\sqrt{2}}\right)(\sigma_r^* - \sigma_{r+1}^*)$ then

$$\boldsymbol{SD}_F(\boldsymbol{U}, \boldsymbol{U}^*) \le \frac{2\sqrt{r}\|\boldsymbol{\Phi} - \boldsymbol{\Phi}^*\|}{\sigma_r^* - \sigma_{r+1}^*}$$

Suppose that, for all $\ell \in \mathcal{J}_{good}$,

$$\Pr\{\|\boldsymbol{\Phi}_\ell - \boldsymbol{\Phi}^*\| \le b_0\} \ge 1-p$$

Using Claim 2, if $b_0 < (1 - 1/\sqrt{2})\Delta$, this implies that, for all $\ell \in \mathcal{J}_{good}$, w.p. at least $1-p$,

$$\boldsymbol{SD}_F(\boldsymbol{U}_\ell, \boldsymbol{U}^*) \le \frac{2\sqrt{r}b_0}{\Delta}$$

Using Lemma 1 with $\delta \equiv \frac{2\sqrt{r}b_0}{\Delta}$, this then implies that, w.p. at least $1 - c_{approxGM} - \exp(-L\psi(0.4-\tau, p))$,

$$\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \le 23\frac{2\sqrt{r}b_0}{\Delta} = 46\sqrt{r}\frac{b_0}{\Delta}$$

To get the right hand side $\le \epsilon$ we need $b_0 \le \frac{\epsilon}{46\sqrt{r}}\Delta$.

D. Proof of Theorem 3.1: SVD at Nodes Computed Using Power Method

This proof also needs to use Claim 3 given below (this is [63, Theorem 1.1]) that analyzes each iteration of what the author calls "noisy power method" (power method that is perturbed by a noise/perturbation $\boldsymbol{G}_t$ in each iteration t.

Claim 3: [Noisy Power Method [63]] Let $\boldsymbol{U}^*$ $(n\times r)$ denote top r singular vectors of a symmetric $n \times n$ matrix $\boldsymbol{\Phi}^*$, and let σ_i denote it's $i-$th singular value. Consider the following algorithm (noisy PM).

1) Let $\boldsymbol{U}_{rand}$ be an $n\times r$ matrix with i.i.d. standard Gaussian entries. Set $\boldsymbol{U}_{t=0} = \boldsymbol{U}_{rand}$.
2) For $t = 1$ to T_{pow} do,
   a) $\hat{\boldsymbol{U}}_t \leftarrow \boldsymbol{\Phi}^* \boldsymbol{U}_{t-1} + \boldsymbol{G}_t$
   b) $\hat{\boldsymbol{U}}_t \leftarrow QR(\hat{\boldsymbol{U}}_t)$

If at every step of this algorithm, we have

$$\begin{aligned}&5\|\boldsymbol{G}_t\| \le \epsilon_{pow}(\sigma_r^* - \sigma_{r+1}^*), \\ &5\|\boldsymbol{U}^{*\top}\boldsymbol{G}_t\| \le (\sigma_r^* - \sigma_{r+1}^*)\frac{\sqrt{r} - \sqrt{r-1}}{\gamma\sqrt{n}}\end{aligned}$$

for some fixed parameter γ and $\epsilon_{pow} < 1/2$. Then w.p. at least $1 - \gamma^{-C_1} - \exp^{-C_2 n}$, there exists a $T_{pow} \geq C\frac{\sigma_r^*}{\sigma_r^* - \sigma_{r+1}^*}\log(\frac{n\gamma}{\epsilon_{pow}})$ so that after T_{pow} steps we have that

$$\|(I - \boldsymbol{U}_{T_{pow}}\boldsymbol{U}_{T_{pow}}^\top)\boldsymbol{U}^*\| \leq \epsilon_{pow}$$

We state below a lower bound on $\sqrt{r} - \sqrt{r-1}$ based on Bernoulli's inequality.

Fact 1: Writing $\sqrt{r} - \sqrt{r-1} = \sqrt{r}\left(1 - \sqrt{1 - \frac{1}{r}}\right)$ and using Bernoulli's inequality $(1+r)^x \leq 1 + xr$ for every real number $0 \leq x \leq 1$ and $r \geq -1$ we have $\frac{1}{2\sqrt{r}} < \sqrt{r} - \sqrt{r-1}$

Suppose that, for all $\ell \in \mathcal{J}_{good}$,

$$\Pr\{\|\boldsymbol{\Phi}_\ell - \boldsymbol{\Phi}^*\| \leq b_0\} \geq 1 - p$$

Using Claim 2, if $b_0 < (1 - 1/\sqrt{2})\Delta$, this implies that, for all $\ell \in \mathcal{J}_{good}$, w.p. at least $1 - p$,

$$\boldsymbol{SD}_F(\boldsymbol{U}_\ell, \boldsymbol{U}^*) \leq \frac{2\sqrt{r}b_0}{\Delta} \tag{5}$$

Suppose that $\hat{\boldsymbol{U}}_\ell$ is an estimate of $\boldsymbol{U}_\ell$ computed using the power method. Next we use Claim 3 to help guarantee that $\boldsymbol{SD}_F(\hat{\boldsymbol{U}}_\ell, \boldsymbol{U}_l)$ is also bounded by $2\sqrt{r}b_0/\Delta$. Using Claim 3 with $\boldsymbol{\Phi} = \boldsymbol{\Phi}_\ell$, $\boldsymbol{U} = \boldsymbol{U}_\ell$, $\boldsymbol{G}_\tau = 0$ for all τ, $\epsilon_{pow} = \frac{2b_0}{\Delta}$, and $\gamma = n^{10}$, we can conclude that if $T_{pow} > C\frac{\sigma_r(\boldsymbol{\Phi}_\ell)}{\sigma_r(\boldsymbol{\Phi}_\ell) - \sigma_{r+1}(\boldsymbol{\Phi}_\ell)}\log(\frac{n \cdot n^{10}}{\epsilon_{pow}})$, then $\boldsymbol{SD}_2(\hat{\boldsymbol{U}}_\ell, \boldsymbol{U}_\ell) \leq \epsilon_{pow} = \frac{2b_0}{\Delta}$ w.p. at least $1 - p - 1/n^{10}$. Here $\sigma_i = \sigma_i(\boldsymbol{\Phi}_\ell)$. Using $\|\hat{\boldsymbol{\Phi}}_\ell - \boldsymbol{\Phi}^*\| \leq b_0$ and Weyl's inequality, $\sigma_r - \sigma_{r+1} \geq \Delta - 2\ b_0$ and $\sigma_r < \sigma_r^* + b_0$. Thus, if

$$T_{pow} \geq C\frac{\sigma_r^* + b_0}{\Delta - 2b_0}\log(n\frac{\Delta}{b_0})$$

then

$$\boldsymbol{SD}_2(\hat{\boldsymbol{U}}_\ell, \boldsymbol{U}_\ell) \leq \epsilon_{pow} = \frac{2b_0}{\Delta}$$

w.p. at least $1 - p - 1/n^{10}$. This then implies that $\boldsymbol{SD}_F(\hat{\boldsymbol{U}}_\ell, \boldsymbol{U}_\ell) \leq \frac{2b_0\sqrt{r}}{\Delta}$.

Combining this bound with the Davis-Kahan bound from (5), we can conclude that, w.p. at least $1 - p - 1/n^{10}$,

$$\boldsymbol{SD}_F(\hat{\boldsymbol{U}}_\ell, \boldsymbol{U}^*) \leq 2\frac{2\sqrt{r}b_0}{\Delta} = 4\sqrt{r}\frac{b_0}{\Delta} \tag{6}$$

Applying Lemma 1 with $\delta \equiv 4\sqrt{r}\frac{b_0}{\Delta}$, this then implies that, w.p. at least $1 - \exp(-L\psi(0.4 - \tau, p + 1/n^{10}))$,

$$\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \leq 23 \cdot 4\sqrt{r}\frac{b_0}{\Delta} = 92\sqrt{r}\frac{b_0}{\Delta} \tag{7}$$

If we want the RHS of the above to be $\leq \epsilon$, we need

$$b_0 = \frac{\epsilon}{92\sqrt{r}}\Delta$$

and we need $T_{pow} \geq C\frac{\sigma_r^* + b_0}{\Delta - 2b_0}\log(n\frac{\Delta}{b_0})$ with this choice of b_0. By substituting for b_0 in the above expression, and upper bounding to simplify it, we get the following as one valid choice of T_{pow}

$$T_{pow} = C(1 + 6\epsilon)\frac{\sigma_r^*}{\Delta}\log(n\frac{92\sqrt{r}}{\epsilon})$$

This used $(1+\epsilon)(1-2\epsilon)^{-1} < (1+\epsilon)(1+4\epsilon) < 1 + 6\epsilon$ for $\epsilon < 1$. Since we are using C to include all constants, and using $\epsilon < 1$, this further simplifies to $T_{pow} = C\frac{\sigma_r^*}{\Delta}\log(\frac{nr}{\epsilon})$

E. Proof of Theorem 3.2

We use Claim 3 with $\boldsymbol{G}_t = \boldsymbol{\Phi}^*\boldsymbol{U} - GM\{\boldsymbol{\Phi}_\ell\boldsymbol{U}\}_{\ell=1}^L$ and output $\boldsymbol{U}_{T_{pow}} \in \Re^{n \times r}$. To apply it, we need $\|\boldsymbol{G}_t\|$ to satisfy the two bounds given in the claim. We use Lemma 6 to bound it.

Suppose that, for at least $(1 - \tau)\, L$, $\boldsymbol{\Phi}_\ell$'s,

$$\Pr\{\|\boldsymbol{\Phi}_\ell - \boldsymbol{\Phi}^*\| \leq b_0\sigma_1^*\} \geq 1 - p$$

Since $\|\boldsymbol{U}\|_F = \sqrt{r}$, this implies

$$\Pr\{\|\boldsymbol{\Phi}_\ell\boldsymbol{U} - \boldsymbol{\Phi}^*\boldsymbol{U}\|_F \leq b_0\sqrt{r}\sigma_1^*\} \geq 1 - p.$$

We use this and apply Lemma 6 with $\boldsymbol{z}_\ell \equiv vec(\boldsymbol{\Phi}_\ell\boldsymbol{U})$ and $\tilde{\boldsymbol{z}} \equiv vec(\boldsymbol{\Phi}^*\boldsymbol{U})$ so that $\|\tilde{\boldsymbol{z}}\| = \|\boldsymbol{\Phi}^*\boldsymbol{U}\|_F \leq \sigma_1^*\sqrt{r}$. Setting $\epsilon_{GM} = b_0$ and applying the lemma, we have w.p. at least $1 - c_{approxGM} - Lp - \exp\left(-L\psi\left(0.4 - \tau, p\right)\right)$

$$\|\boldsymbol{G}_t\| \leq \|\boldsymbol{G}_t\|_F = \|GM\{\boldsymbol{\Phi}_\ell\boldsymbol{U}\}_{\ell=1}^L - \boldsymbol{\Phi}^*\boldsymbol{U}\|_F \leq 14b_0\sqrt{r}\sigma_1^*$$

Recall that $\sigma_r^* - \sigma_{r+1}^* \geq \Delta$. We thus need $5\|\boldsymbol{G}_t\| \leq \epsilon\Delta$ to hold. This will hold with high probability if $b_0\sqrt{r}\sigma_1^* \leq \frac{\epsilon\Delta}{70}$. Using Fact 1 and $\gamma = c$, for the second condition of Claim 3 to hold, we need $\|\boldsymbol{G}_t\| \leq \Delta\frac{1}{10c\sqrt{nr}}$. This then implies that we need $b_0\sqrt{r}\sigma_1^* \leq \frac{\Delta}{140c\sqrt{nr}}$.

Thus we can set $b_0 = \min\left(\frac{\epsilon}{70\sqrt{r}}, \frac{1}{140c\sqrt{nr}}\right)\frac{\Delta}{\sigma_1^*}$.

We also need $T_{pow} > C\frac{\sigma_r^*}{\sigma_r^* - \sigma_{r+1}^*}\log(\frac{n\gamma}{\epsilon})$. This holds if we set $T_{pow} = C\frac{\sigma_r^*}{\Delta}\log(\frac{nc}{\epsilon})$.

Hence w.p. at least $1 - c_{approxGM}Lp - \exp\left(-L\psi\left(0.4 - \tau, p\right)\right) - c - e^{-C_2 n} \geq 1 - c_{approxGM} - c - Lp - \exp\left(-L\psi\left(0.4 - \tau, p\right)\right)$

$$\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \leq \epsilon$$

F. Proof of Corollary 1

The first part is a corollary of Theorem 3.1 and [56, Theorem 4.7.1] stated next. It gives a high probability bound on the error between an empirical covariance matrix estimate, $\hat{\boldsymbol{\Sigma}} = \boldsymbol{D}\boldsymbol{D}^\top/\widetilde{q}$, with the $\widetilde{q}$ columns of $\boldsymbol{D}$ being independent sub-Gaussian random vectors $\boldsymbol{d}_k$, and the true one, $\boldsymbol{\Sigma}^*$.

Claim 4 ([56]): Suppose that the matrix $\boldsymbol{D}$ is as defined in Sec. IV. With probability at least $1 - 2\exp(-n)$,

$$\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}^*\| \leq CK^2\sqrt{\frac{n}{\widetilde{q}}}\|\boldsymbol{\Sigma}^*\|.$$

Here K is the maximum sub-Gaussian norm of $\boldsymbol{\Sigma}^{*-1/2}\boldsymbol{d}_k$ over k.

Using Theorem 3.1 with $\boldsymbol{\Phi}_\ell \equiv \hat{\boldsymbol{\Sigma}}_\ell = \boldsymbol{D}_\ell\boldsymbol{D}_\ell^\top/\widetilde{q}$, $\boldsymbol{\Phi}^* \equiv \boldsymbol{\Sigma}^*$, in order to guarantee $\boldsymbol{SD}(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \leq \epsilon$ w.h.p., we need

$$\|\hat{\boldsymbol{\Sigma}}_\ell - \boldsymbol{\Sigma}^*\| \leq \frac{\epsilon\Delta}{92\sqrt{r}}$$

By Claim 4,

$$\Pr\{\|\hat{\boldsymbol{\Sigma}}_\ell - \boldsymbol{\Sigma}^*\| \leq CK^2\sqrt{\frac{n}{\widetilde{q}}}\|\boldsymbol{\Sigma}^*\|\} \geq 1 - 2\exp(-n)$$

The above bound will be less than $\frac{\epsilon\Delta}{92\sqrt{r}}$ if $\widetilde{q} \geq \frac{92^2 CK^4 nr\|\boldsymbol{\Sigma}^*\|^2}{\Delta^2\epsilon^2}$.

G. Proof of Corollary 2

Corollary 2 is again a direct corollary of Theorem 3.1 and [56, Theorem 4.7.1]. We now apply both results on $\boldsymbol{\Phi}_{(\vartheta)} := \sum_{\ell=1}^{\rho} \boldsymbol{D}_{(\vartheta,\ell)} \boldsymbol{D}_{(\vartheta,\ell)}^{\top}/(\widetilde{q}\rho)$, $\vartheta \in [\tilde{L}]$. The reason this proof follows exactly as that for subspace median is because we assume that the set of Byzantine nodes is fixed across all iterations of this algorithm and the number of such nodes is lower by a factor of $L/\tilde{L}$. Consequently, for the purpose of the proof one can assume that no more than $\tau\tilde{L}$ mini-batches are Byzantine. With this, the proof remains the same once we replace $\widetilde{q}$ by $\widetilde{q}\rho$ and L by $\tilde{L}$.

VII. Simulation Experiments

All numerical experiments were performed using MATLAB on Intel(R)Xeon(R) CPU E3-1240 v5 @ 3.50GHz processor with 32.0 GB RAM.

A. PCA Experiments

1) Data Generation: We generated $\boldsymbol{\Phi}^* = \boldsymbol{U}_{full}^* \boldsymbol{S}_{full} \boldsymbol{U}_{full}^{*\top}$, with $\boldsymbol{U}_{full}^*$ generated by orthogonalizing an $n \times n$ standard Gaussian matrix; $\boldsymbol{S}_{full}$ is a diagonal matrix of singular values which are set as described below. This was generated once. The model parameters $n,\ r, q,\ L, L_{byz}$, and entries of $\boldsymbol{S}_{full}$ are set as described below in each experiment.

In all our experiments in this section, we averaged over 1000 Monte Carlo runs. In each run, we sampled q vectors from the Gaussian distribution, $\mathcal{N}(\mathbf{0}, \boldsymbol{\Phi}^*)$ to form the data matrix $\boldsymbol{D}$. This is split into L columb sub-matrices, $\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_L$ with each containing $\widetilde{q} = q/L$ columns. q, L are set so that q/L is an integer. Each run also generated a new $\boldsymbol{U}_{rand}$ to initialize the power method used by the nodes in case of SubsMed and used by the center in case of ResPowMeth. The same one was also used by the power methods for SubsMoM. Note: since SubsMed and SubsMoM run and $\tilde{L}$ different power methods, ideally each could use a different $\boldsymbol{U}_r and$ and that would actually improve their performance. To be fair to all three methods, we generated U_{rand} this way.

Let $L_{byz} = \tau L$. In all our experiments, we fixed $n = 1000$ and varied $r, q,\ L, L_{byz}$, and $\boldsymbol{S}_{full}$. In all experiments we used a large singular value gap (this ensures that a small value of T_{pow} suffices). We experimented with three types of attacks described next.

2) Attacks: To our best knowledge, the PCA problem has not been studied for Byzantine resiliency, and hence, there are no known difficult attacks for it. It is impossible to simulate the most general Byzantine attack. We focused on three types of attacks. Motivated by reverse gradient (rev) attack [66], we generated the first one by colluding with other nodes to set $\boldsymbol{U}_{corrupt}$ as a matrix in the subspace orthogonal to that spanned by $\sum_\ell \hat{\boldsymbol{U}}_\ell$ at each iteration. This is generated as follows. Let $\boldsymbol{U} = \sum_\ell \hat{\boldsymbol{U}}_\ell$ (in case of SubsMed, SubsMoM) and $\boldsymbol{U} = \sum_\ell \boldsymbol{\Phi}_\ell \boldsymbol{U}_t$ (for ResPowMeth). Orthonormalize it $\tilde{\boldsymbol{U}} = orth(\boldsymbol{U})$ and let $\tilde{\boldsymbol{M}} = I - \tilde{\boldsymbol{U}}\tilde{\boldsymbol{U}}^{\top}$, obtain its QR decomposition $\tilde{\boldsymbol{M}} \overset{QR}{=} \boldsymbol{U}_{perp} R$ and set $\boldsymbol{U}_{corrupt} = (\omega_{GM}/\sqrt{r})\boldsymbol{U}_{perp}(:, 1:r)$. We call this *Orthogonal attack*. Since SubsMed runs all its iterations locally, this is generated once for SubsMed, but it is generated at each iteration for ResPowMeth and SubsMoM.

The second attack that we call the *ones attack* consists of an $n \times r$ matrix of -1 multiplied by a large constant C_{attack}. The third attack that we call the *Alternating attack* is an $n \times r$ matrix of alternating $+1$, -1 multiplied by a large constant $C_{attack} > 0$. Values of C_{attack} were chosen so that they do not get filtered out, essentially $0.9\omega_{GM}/\sqrt{nr}$.

3) Algorithm Parameters: For all geometric median (GM) computations, we used Weiszfeld's algorithm initialized using the average of the input data points. We set $T_{GM} = 10$. We vary T_{pow}.

4) Experiments: In all experiments, we compare ResPowMeth and SubsMed. In some of them, we also compare SubsMoM. We also report results for the basic power method in the no attack setting. To provide a baseline for what error can be achieved for a given value of n, q, r, we also report results for using "standard power method" in the no-attack setting; with this being implemented using power method with T_{pow} iterations. *Our reporting format is "*$\max \boldsymbol{SD}_F(\textit{mean}\boldsymbol{SD}_F), \textit{mean time}$*" in the first table and just "*$\max \boldsymbol{SD}_F(\textit{mean}\boldsymbol{SD}_F)$*" in the others.* Here $\max \boldsymbol{SD}_F$ is the worst case error over all 1000 Monte Carlo runs, while mean$\boldsymbol{SD}_F$ is its mean over the runs.

In our first experiment, we let $n = 1000$, $r = 60$, $q = 1800$, $L = 3, L_{byz} = 1$, and we let $\boldsymbol{S}_{full}$ be a full rank diagonal matrix with first r entries set to 15, the $r+1$-th entry to 1, and the others generated as $1 - (1/n), 1 - (2/n), \ldots$. Next, we simulated an approximately low rank $\boldsymbol{\Sigma}^*$ by setting its first r entries set to 15, the $r+1$-th entry to 1, and the other entries to zero. We report results for both these experiments in Table II. As can be seen, from the first to the second experiment, the error reduces for both SubsMed and ResPowMeth, but the reduction is much higher for SubsMed. Notice also that, for $Tpow = 1$, both ResPowMeth and SubsMed have similar and large errors with that of SubsMed being very marginally smaller. For $T_{pow} = 10$, SubsMed has significantly smaller errors than ResPowMeth for reasons explained in the paper. ResPowMeth has lower errors for the Orthogonal attack than for the other two; we believe the reason is that the Orthogonal attack changes at each iteration for ResPowMeth.

We also did some more experiments with (i) $L = 3, L_{byz} = 1$, $r = 2, q = 360$, (ii) $L = 6, L_{byz} = 2$, $r = 2, q = 720$, and (iii) $L = 6, L_{byz} = 2$, $r = 60, q = 3600$. All these results are reported in Table III. Similar trends to the above are observed for these too.

In a third set of experiments, we used $L = 18$, $r = 60, q = 3600$ and two values of L_{byz}, $L_{byz} = 2, L_{byz} = 4$. For this one, we also compared SubsMoM with using $\tilde{L} = 6$ minibatches. In the $L_{byz} = 2$ case, SubsMoM has the smallest errors, followed by SubsMed. Error of ResPowMeth is the largest. In the $L_{byz} = 4$ case, ResPowMeth still has the largest errors. But in this case SubsMoM with $\tilde{L} = 6$ also fails (when taking the GM of 6 points, 4 corrupted points is too large. SubsMed has the smallest errors in this case. We report results for these experiments in Table IV

TABLE II

$n = 1000$, $L = 3$, $L_{byz} = 1$, $r = 60$, $q = 1800$. WE REPORT "$\max \boldsymbol{SD}_F(\text{MEAN}\boldsymbol{SD}_F)$, MEAN TIME" IN EACH COLUMN

Attacks	Methods	$T_{pow} = 10$
Alternating	**SubsMed**	0.375(0.348),0.680
	ResPowMeth	1.000(0.972),0.475
Ones	**SubsMed**	0.369(0.349),0.704
	ResPowMeth	0.999(0.990),0.513
Orthogonal	**SubsMed**	0.365(0.348),0.689
	ResPowMeth	0.999(0.366),0.500
No Attack	**Power(Baseline)**	0.187(0.182),0.529

(a) full rank $\boldsymbol{\Sigma}^*$

Attacks	Methods	$T_{pow} = 10$	$T_{pow} = 1$
Alternating	**SubsMed**	0.110(0.091),0.689	0.999(0.614),0.326
	ResPowMeth	0.971(0.898),0.497	1.000(0.991),0.049
Ones	**SubsMed**	0.111(0.091),0.669	0.999(0.607),0.331
	ResPowMeth	0.992(0.952),0.477	0.999(0.990),0.052
Orthogonal	**SubsMed**	0.106(0.091),0.672	0.999(0.609),0.319
	ResPowMeth	0.223(0.208),0.475	0.999(0.993),0.048
No Attack	**Power(Baseline)**	0.063(0.050),0.505	0.999(0.605),0.050

(b) rank-$(r+1)$ $\boldsymbol{\Sigma}^*$

TABLE III

ADDITIONAL EXPERIMENTS. WE REPORT "$\max \boldsymbol{SD}_F(\text{MEAN}\boldsymbol{SD}_F)$" IN EACH COLUMN

Attacks	Methods	$T_{pow} = 10$	$T_{pow} = 1$
Alternating	**SubsMed**	0.062(0.030)	0.997(0.289)
	ResPowMeth	0.177(0.084)	0.999(0.424)
Ones	**SubsMed**	0.080(0.030)	0.989(0.236)
	ResPowMeth	0.196(0.087)	0.972(0.311)
Orthogonal	**SubsMed**	0.067(0.033)	0.999(0.228)
	ResPowMeth	0.125(0.066)	0.999(0.375)
No Attack	**Power(Baseline)**	0.038(0.018)	0.968(0.211)

(a) $L = 3, L_{byz} = 1,\ r = 2, q = 360$

Attacks	Methods	$T_{pow} = 10$	$T_{pow} = 1$
Alternating	**SubsMed**	0.045(0.020)	0.986(0.227)
	ResPowMeth	0.178(0.080)	0.997(0.336)
Ones	**SubsMed**	0.048(0.020)	0.999(0.275)
	ResPowMeth	0.157(0.081)	0.999(0.383)
Orthogonal	**SubsMed**	0.049(0.019)	0.999(0.204)
	ResPowMeth	0.102(0.057)	0.999(0.339)
No Attack	**Power**	0.033(0.012)	0.975(0.203)

(b) $L = 6, L_{byz} = 2,\ r = 2, q = 720$

Attacks	Methods	$T_{pow} = 10$	$T_{pow} = 1$
Alternating	**SubsMed**	0.098(0.085)	0.999(0.642)
	ResPowMeth	0.992(0.853)	1.000(0.988)
Ones	**SubsMed**	0.099(0.084)	0.999(0.625)
	ResPowMeth	0.998(0.905)	0.999(0.989)
Orthogonal	**SubsMed**	0.103(0.084)	0.999(0.610)
	ResPowMeth	0.223(0.184)	0.999(0.993)
No Attack	**Power**	0.043(0.036)	0.995(0.604)

(c) $L = 6, L_{byz} = 2,\ r = 60, q = 3600$

TABLE IV

$L = 18$, RANK-$(r+1)$ $\boldsymbol{\Sigma}^*$, $r = 60$, $q = 3600$, $T_{GM} = 10$, $\tilde{L} = 6$ FOR SUBSMOM.WE REPORT "$\max \boldsymbol{SD}_F(\text{MEAN}\boldsymbol{SD}_F)$" IN EACH COLUMN

Attacks	Methods	$T_{pow} = 10$
Alternating	**SubsMoM**	0.101(0.085)
	SubsMed	0.175(0.150)
	ResPowMeth	0.522(0.463)
Ones	**SubsMoM**	0.098(0.085)
	SubsMed	0.172(0.150)
	ResPowMeth	0.542(0.502)
Orthogonal	**SubsMoM**	0.104(0.086)
	SubsMed	0.172(0.151)
	ResPowMeth	0.223(0.191)
No Attack	**Power(Baseline)**	0.042(0.035)

(a) $L_{byz} = 2$

Attacks	Methods	$T_{pow} = 10$
Alternating	**SubsMoM**	0.999(0.872)
	SubsMed	0.182(0.152)
	ResPowMeth	0.987(0.894)
Ones	**SubsMoM**	1.000(1.000)
	SubsMed	0.160(0.147)
	ResPowMeth	0.999(0.948)
Orthogonal	**SubsMoM**	1.000(1.000)
	SubsMed	0.183(0.151)
	ResPowMeth	0.216(0.179)
No Attack	**Power(Baseline)**	0.040(0.036)

(b) $L_{byz} = 4$

B. LRCS Experiments

In all experiments, we used $n = 600$, $q = 600$, $r = 4$, $m = 198$, and $L = 18$ so that $\widetilde{m} = m/L = 11$ and two values of $L_{byz} = 1, 2$. We simulated $\boldsymbol{U}^*$ by orthogonalizing an $n \times r$ standard Gaussian matrix; and the columns $\boldsymbol{b}_k^*$ were generated i.i.d. from $\mathcal{N}(0, I_r)$. We then set $\boldsymbol{X}^* = \boldsymbol{U}^*\boldsymbol{B}^*$. This was done once (outside Monte Carlo loop). For 100 Monte Carlo runs, we generated matrices $\boldsymbol{A}_k, k \in [q]$ with each entry being i.i.d. standard Gaussian and we set $\boldsymbol{y}_k = \boldsymbol{A}_k\boldsymbol{x}_k^*$, $k \in [q]$. In the figures we plot Error vs Iteration where $Error = \frac{\boldsymbol{SD}_F(\boldsymbol{U}^*,\boldsymbol{U})}{\sqrt{r}}$. We simulated the Reverse gradient (Rev) attack for the gradient step. In this case, malicious gradients are obtained by finding the empirical mean of the gradients from all nodes: $\nabla \leftarrow \frac{1}{L}\sum_{\ell=1}^{L}\nabla_\ell$ and set $\nabla_{mal} = -C\nabla$ where $C = 10$. This forces the GD step to move in the reverse direction of the true gradient. We used step size $\eta = \frac{0.5}{\sigma_1^{*2}}$. We used Weiszfeld's method to approximately compute geometric median.

Method	$L_{byz}=1$	$L_{byz}=2$
SubsMed	0.716(0.665)	0.717(0.667)
SubsMoM	0.477(0.457)	0.475(0.459)

(a) Initialization errors

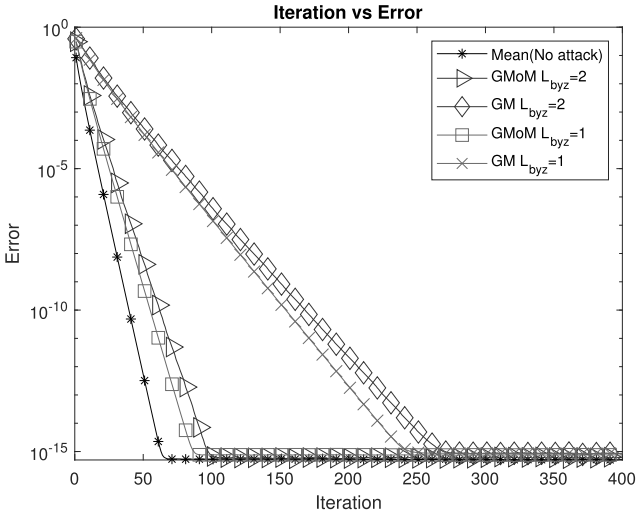

Fig. 1. Byz-AltGDmin (Median) vs Byz-AltGDmin (MoM) for $L_{byz}=1,2$; $L=18$.

We compare Byz-AltGDmin (Median) with Byz-AltGDmin (MoM) for both values of L_{byz}. We also provide results for the baseline algorithm - basic AltGDmin in the no attack setting. All these are compared in Figure 1(b). We also compare the initialization errors in Figure 1(a). As can be seen SubsMoM based initialization error is quite a bit lower than that with SubsMed. The same is true for the GDmin iterations.

C. How to Tune or Set Parameters for a Real Application

Consider the PCA problem and Subspace Median. Suppose that the user of the algorithm specifies the desired dimension r and the desired final estimation error ϵ. Our theoretical guarantee specifies that we need $T_{pow} = C\frac{\sigma_r^*}{\Delta}\log(\frac{n}{\epsilon})$, and $T_{GM} = C\log(Lr/\epsilon)$. Given r and ϵ (desired error), for setting T_{GM} all values are known. For T_{pow} we need σ_r^* and Δ. To estimate these, at each node ℓ, we compute the r-th and $(r+1)$-th singular values of $\mathbf{\Phi}_\ell = \boldsymbol{D}_\ell \boldsymbol{D}_\ell^\top/q_\ell$. Denote these by $\hat{\sigma}_{\ell,r}$, and $\hat{\Delta}_\ell = \hat{\sigma}_{\ell,r} - \hat{\sigma}_{\ell,r+1}$. We use $\max_\ell\{\hat{\sigma}_{\ell,r}\}$, $\min_\ell\{\hat{\Delta}_\ell\}$ in T_{pow}. The constants C in various expressions will typically need to be experimentally tuned for a given application. It should be noted that sufficiently large values of T_{GM}, T_{pow}, e.g., setting both t0 10, works well for all algorithms without much change in final error. If the user does not specify r we can set r using the well known 90 or 99% energy threshold heuristic. We find r_ℓ as the smallest value of r for which the sum of the top r singular values is at least 90% (or 99% or similar) of the sum of all singular values of $\mathbf{\Phi}_\ell$. Instead of setting ϵ and T_{pow}, we can set a stopping criterion for the power method being implemented at each node: exit the loop if the estimates do not change much in subspace distance.

Consider LRCS. For Byz-AltGDmin we set ω_{GM} as $\tilde{m}14\sqrt{r}\delta_0 \max_\ell\{\hat{\sigma}_{\ell,1}\}$. The idea is to set the threshold ω_{GM} sufficiently large to ensure that non-Byzantine updates are not filtered out. For other parameters in Byz-AltGDmin $\tilde{C} = 9\kappa^2\mu^2$, we set $\kappa = \max_\ell\{\kappa_\ell\}$, and take $\mu \geq 2$. We set $T = C\max_\ell\{\kappa_\ell^2\}\log(1/\epsilon)$, and $\eta = \frac{0.5}{\max_\ell\{\hat{\sigma}_{\ell,1}\}}$.

VIII. Conclusion, Extensions, and Open Questions

Our work introduced a novel and well-motivated solution to Byzantine-resilient federated subspace estimation, and PCA, that is both communication-efficient and sample-efficient. We refered to this as "Subspace-Median". Its guarantee is provided in Theorem 3.1 and Corollary 1. We showed how the Subspace Median can be used to provably solve two practically useful problems: (i) Byzantine resilient federated PCA, and (ii) the initialization step of Byzantine-resilient horizontal federated LRCS. We also developed Subspace Median-of-Means (MoM) extensions for both problems. These help improve the sample complexity at the cost of reduced Byzantine/outlier tolerance. For all these algorithms, Theorem 3.1 helps prove sample, communication, and time complexity bounds for ϵ-accurate subspace recovery. Extensive simulation experiments corroborate our theoretical results. Our second important contribution is a provable communication-efficient and sample-efficient alternating GD and minimization (altGDmin) based solution to horizontally federated LRCS, obtained by using the Subspace Median to initialize the alternating GD and minimization (altGDmin) algorithm for solving it. Our proposed algorithms and proof techniques are likely to be of independent interest for many other problems. We describe some extensions next.

A. Extensions

One component that is missing in most existing work on Byzantine resilient federated GD, and stochastic GD, based solutions is how to initialize the GD algorithm in such a way that the problem becomes restricted strongly convex in the vicinity of the desired/true solution. Most existing works either assume strongly convex cost functions or prove convergence to a local minimizer of a cost function. However, good initialization of the GD algorithm is a critical component for correctly solving a large number of practical problems. The spectral initialization approach has been extensively used for developing provably correct centralized iterative solutions to many non-convex optimization problems in signal processing and ML. It involves computing the top, or top few, singular vectors of an appropriately defined matrix. This can be made Byzantine resilient and communication-efficient in a federated setting, by using the Subspace Median and Subspace MoM algorithms introduced in this work. Examples include

LRCS, LR matrix completion, robust PCA using the LR plus sparse model, phase retrieval (PR), sparse PR, and low rank PR.

The overall approach developed here for modifying the altGDmin algorithm can also be widely used in other settings. In particular, vertically federated LRCS can be analyzed using easy extensions of our current work. It would require assuming that the Frobenius norm of the difference between column-sub matrices of $\boldsymbol{X}^*$ that are sensed at the different nodes is bounded. This assumption is needed to ensure bounded heterogeneity of the different nodes' partial gradient estimates; a common assumption used in all past work on federated ML with heterogeneous nodes. Similar ideas can also extend for LR matrix completion, which also involves dealing with heterogeneous gradients. Vertically federated LRCS is the model that is relevant for federated sketching, and for multi-task representation learning when data for different tasks is obtained at different nodes.

Our Byzantine resilient PCA result can be generalized to extend it to various other PCA-based problems. Some examples are described in Remark 4 (PCA for non-i.i.d. data, PCA for approximately LR datasets, PCA with missing data). Other examples include online PCA, subspace tracking, robust subspace tracking, differentially private PCA [63].

B. Open Questions

In the current work, we treated the geometric median computation as a black box. Both for its accuracy and its time complexity, we relied on results from existing work. However, notice that in the Subspace Median algorithm (which is used in all other algorithms in this work), we need a "median" of r-dimensional subspaces in $\Re^n$. These are represented by their $n \times r$ basis matrices. To find this though, we are computing the geometric median (GM) of vectorized versions of the subspace projection matrices $\mathcal{P}_{\boldsymbol{U}} := \boldsymbol{U}\boldsymbol{U}^\top$ which are of size $n \times n$. Eventually we need to find the subspace whose projection matrix is closest to the GM. An open question is can we develop a more efficient algorithm to do this computation that avoids having to compute the GM of n^2 length vectors. We will explore the use of power method type ideas for modifying the GM computation algorithm in order to do this. Alternatively, can we define a different notion of "median" for subspaces that can be computed more efficiently than Subspace Median. Another related question is whether the computation can be federated to utilize the parallel computation power of the various nodes. In its current form, the entire GM computation is being done at the center. A third open question is whether we can improve the guarantees for the Subspace Median of Means algorithms by using more sophisticated proof techniques, such as those used in [8].

APPENDIX A
PROOF OF LRCS INITIALIZATION, COROLLARY 3

We prove this result for the $\tilde{L} = L$ setting below. The extension for the $\tilde{L} < L$ setting is straightforward. We explain this in Appendix A-C below.

A. Lemmas for Proving Corollary 3 for $\tilde{L} = L$

We first state the lemmas from [49] that are used in the proof and then provide the proof.

Lemma 5: Define the set

$$\mathcal{E} := \left\{ \tilde{C}(1-\epsilon_1)\frac{\|\boldsymbol{X}^*\|_F^2}{q} \le \alpha_\ell \le \tilde{C}(1+\epsilon_1)\frac{\|\boldsymbol{X}^*\|_F^2}{q} \right\}$$

then $\Pr(\alpha \in \mathcal{E}) \ge 1 - L\exp(-\tilde{c}\widetilde{m}q\epsilon_1^2)$ where $\alpha = \text{Median}\{\alpha_\ell\}_{\ell=1}^L$

Proof: Threshold computation: From [49] Fact 3.7 for all $\ell \in \mathcal{J}_{good}$

$$\Pr\{\alpha_\ell \in \mathcal{E}\} \ge 1 - \exp(-\tilde{c}\widetilde{m}q\epsilon_1^2)$$

Since more than 75% of α_ℓ's are *good* and the median is same as the 50th percentile for a set of scalars. This then implies Median$\{\alpha_\ell\}_{\ell=1}^L$ will be upper and lower bounded by *good* α_ℓ's. Taking union bound over *good* α_ℓ's w.p. at least $1 - L\exp(-\tilde{c}\widetilde{m}q\epsilon_1^2) = 1 - p_\alpha$

$$\Pr\{\alpha \in \mathcal{E}\} \ge 1 - L\exp(-\tilde{c}\widetilde{m}q\epsilon_1^2)$$

□

Lemma 6 ([49]): Define $(\boldsymbol{X}_0)_\ell := \sum_k (\boldsymbol{A}_k)_\ell{}^\top ((\boldsymbol{y}_k)_\ell)_{\text{trunc}} \boldsymbol{e}_k^\top$, $\boldsymbol{y}_{k,trnc} := (\boldsymbol{y}_k \circ \boldsymbol{1}_{|\boldsymbol{y}_k| \le \sqrt{\alpha}})$. Conditioned on α, we have the following conclusions.

1) Let ζ be a scalar standard Gaussian r.v.. Define,

$$\beta_k(\alpha) := \mathbb{E}[\zeta^2 \mathbb{1}_{\{\|\boldsymbol{x}_k^*\|^2\zeta^2 \le \alpha\}}]$$

Then,

$$\mathbb{E}[(\boldsymbol{X}_0)_\ell | \alpha] = \boldsymbol{X}^* \boldsymbol{D}(\alpha)$$

where $\boldsymbol{D}(\alpha) = diagonal(\beta_k(\alpha), k \in [q])$, i.e., $\boldsymbol{D}(\alpha)$ is a diagonal matrix of size $q \times q$ with diagonal entries β_k defined above.

2) Fix $0 < \epsilon_1 < 1$. Then w.p. at least $1 - \exp[(n+q) - c\epsilon_1^2 \widetilde{m}q/\mu^2\kappa^2]$

$$\|(\boldsymbol{X}_0)_\ell - \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\| \le 1.1\epsilon_1 \|\boldsymbol{X}^*\|_F$$

3) For any $\epsilon_1 \le 0.1$, $\min_k \mathbb{E}\left[\zeta^2 \mathbb{1}_{\left\{|\zeta| \le \tilde{C}\frac{\sqrt{1-\epsilon_1}\|\boldsymbol{X}^*\|_F}{\sqrt{q}\|\boldsymbol{x}_k^*\|}\right\}}\right] \ge 0.92$

Fact 2: For any $t > 0$, $\mathbb{E}[\zeta^2 \mathbb{1}_{\{\zeta^2 \le t\}}] \le 1$, this then implies $\|\boldsymbol{D}(\alpha)\| \le 1$

B. Proof of Corollary 3 for $\tilde{L} = L$

We will apply Theorem 3.1 with $\boldsymbol{\Phi}_\ell \equiv (\boldsymbol{X}_0)_\ell (\boldsymbol{X}_0)_\ell^\top$, $\boldsymbol{\Phi}^* \equiv \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]^\top = \boldsymbol{X}^*\boldsymbol{D}(\alpha)^2\boldsymbol{X}^{*\top}$, $\epsilon = \delta_0$ and $\Delta = \sigma_r(\boldsymbol{X}^*\boldsymbol{D}(\alpha)^2\boldsymbol{X}^{*\top}) - \sigma_{r+1}(\boldsymbol{X}^*\boldsymbol{D}(\alpha)^2\boldsymbol{X}^{*\top})$. For this we need to bound $\|(\boldsymbol{X}_0)_\ell(\boldsymbol{X}_0)_\ell^\top - \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]^\top\|$. We can write

$$\begin{aligned}
&(\boldsymbol{X}_0)_\ell(\boldsymbol{X}_0)_\ell^\top - \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]^\top \\
&= (\boldsymbol{X}_0)_\ell((\boldsymbol{X}_0)_\ell - \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha])^\top \\
&+ ((\boldsymbol{X}_0)_\ell - \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha])\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]^\top \qquad (8)
\end{aligned}$$

To bound (8) we need the bounds on $(\boldsymbol{X}_0)_\ell$, $\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]$, and $(\boldsymbol{X}_0)_\ell - \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]$

1) From Lemma 6 part 2, letting $\epsilon_1 = \epsilon_3/\sqrt{r}$, w.p. $1 - \exp[(n+q) - c\epsilon_1^2 \widetilde{m} q/r\mu^2\kappa^2]$

$$\|(\boldsymbol{X}_0)_\ell - \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\| \le 1.1\epsilon_3\sigma_1^* \tag{9}$$

2) From Lemma 6 part 1, Fact 2

$$\|\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\| \le \sigma_1^* \tag{10}$$

3) Thus, for $\epsilon_3 < 0.1$, w.p. $1-\exp[(n+q) - c\epsilon_3^2\widetilde{m}q/r\mu^2\kappa^2]$

$$\begin{aligned}\|(\boldsymbol{X}_0)_\ell\| &= \|(\boldsymbol{X}_0)_\ell - \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\| + \|\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\| \\ &\le 1.1(1+\epsilon_3)\sigma_1^* < 1.3\sigma_1^*\end{aligned} \tag{11}$$

.

From (8), (9),(10) and (11), and $\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha] = \boldsymbol{X}^*\boldsymbol{D}(\alpha)$, we have w.p. at least $1 - 2\exp[(n+q) - c\epsilon_3^2\widetilde{m}q/r\mu^2\kappa^2]$

$$\begin{aligned}&\|(\boldsymbol{X}_0)_\ell(\boldsymbol{X}_0)_\ell^\top - \boldsymbol{X}^*\boldsymbol{D}(\alpha)^2\boldsymbol{X}^{*\top}\| \le \\ &\|(\boldsymbol{X}_0)_\ell\|\|(\boldsymbol{X}_0)_\ell - \boldsymbol{X}^*\boldsymbol{D}(\alpha)\| \\ &+ \|(\boldsymbol{X}_0)_\ell - \boldsymbol{X}^*\boldsymbol{D}(\alpha)\|\|\boldsymbol{D}(\alpha)\boldsymbol{X}^*\| \\ &\le 1.1.1.1\epsilon_3\sigma_1^{*2} + 1.1\epsilon_3\sigma_1^{*2} < 2.5\epsilon_3\sigma_1^{*2}\end{aligned} \tag{12}$$

To apply Theorem 3.1 to get $\boldsymbol{SD}_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) < \delta_0$, we need $\|(\boldsymbol{X}_0)_\ell(\boldsymbol{X}_0)_\ell^\top - \boldsymbol{X}^*\boldsymbol{D}(\alpha)^2\boldsymbol{X}^{*\top}\| < \frac{\delta_0}{26\sqrt{r}}\Delta$. By Lemma 6 part 3, Fact 2, and the fact that $\boldsymbol{X}^*$ is rank r we get a lower bound on Δ

$$\Delta \ge 0.92^2\sigma_r^{*2} - 0 > 0.8\sigma_r^{*2} \tag{13}$$

Using (13) and (12), the required condition for Theorem 3.1 holds if

$$2.5\epsilon_3\sigma_1^{*2} \le 0.8\frac{\delta_0}{26\sqrt{r}}\sigma_r^{*2}$$

This will hold if we set $\epsilon_3 = \frac{c}{\sqrt{r}\kappa^2}\delta_0$. With this choice of ϵ_3, the bounds hold w.p. at least $1-2\exp[(n+q)-c\delta_0^2\widetilde{m}q/r^2\mu^2\kappa^6]$

Thus, by Theorem 3.1,

$$\begin{aligned}&\Pr\{SD_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \le \delta_0|\alpha\} \\ &\ge 1 - c_0 - \exp(-L\psi(0.4-\tau, p + n^{-10}))\end{aligned}$$

where $p = 2\exp[(n+q) - c\delta_0^2\widetilde{m}q/r^2\mu^2\kappa^6]$.

Following the same argument as given in proof of [49, Theorem 3.1] and using Lemma 5 to remove the conditioning on α, we get

$$\begin{aligned}&\Pr\{SD_F(\boldsymbol{U}_{out}, \boldsymbol{U}^*) \le \delta_0\} \\ &\ge 1 - c_0 - \exp(-L\psi(0.4-\tau, p + n^{-10})) - p_\alpha\end{aligned}$$

where $p_\alpha = L\exp(-\tilde{c}\widetilde{m}q\delta_0^2/r^2\kappa^4)$.

If $\widetilde{m}q \ge C\kappa^6\mu^2(n+q)r^2/\delta_0^2$, then $p < e^{-c(n+q)}$, and $p_\alpha < e^{-c(n+q)}$.

Thus, the good event holds w.p. at least $1 - c_0 - \exp(-L\psi(0.4-\tau, e^{-c(n+q)} + n^{-10})) - e^{-c(n+q)}$.

C. Proof of Corollary 3 for a $\tilde{L} < L$

In this case, we apply Theorem 3.1 on $\boldsymbol{\Phi}_{(\vartheta)} = \sum_{\ell=1}^{\rho}(\boldsymbol{X}_0)_{(\vartheta,\ell)}(\boldsymbol{X}_0)_{(\vartheta,\ell)}^\top/(\widetilde{m}\rho)^2$ and $\boldsymbol{\Phi}^* = \mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]\,\mathbb{E}[(\boldsymbol{X}_0)_\ell|\alpha]^\top/\widetilde{m}^2$ for $\vartheta \in [\tilde{L}]$. To obtain the bounds needed to apply Theorem 3.1, we use the bounds from above.

APPENDIX B
PROOFS FOR LRCS ALTGDMIN ITERATIONS

We prove this for the simple GM setting because that is notation-wise simpler. This is the $\tilde{L} = L$ setting. The extension to GMoM is straightforward once again.

All expected values used below are expectations conditioned on past estimates (which are functions of past measurement matrices and measurements, $\boldsymbol{A}_k, \boldsymbol{y}_k$). For example, $\mathbb{E}[(\nabla_U f)_\ell]$ conditions on the values of $\boldsymbol{U}, \boldsymbol{B}_\ell$ used to compute it. This is also the reason why $\mathbb{E}[(\nabla_U f)_\ell]$ is different for different nodes; see Lemma 2.

A. Lemmas for Proving Theorem 5.3 for $\tilde{L} = L$: LS Step Bounds

The next lemma bounds the 2-norm error between $(\boldsymbol{b}_k)_\ell$ and an appropriately rotated version of $\boldsymbol{b}_k^*$, $\boldsymbol{g}_k := \boldsymbol{U}^\top\boldsymbol{x}_k^* = (\boldsymbol{U}^\top\boldsymbol{U}^*)\boldsymbol{b}_k^*$; followed by also proving various important implications of this bound. Here and below $\boldsymbol{U}$ denotes the subspace estimate at iteration t.

Lemma 7 (Lemma 3.3 of [49]): Assume that $\boldsymbol{SD}_F(\boldsymbol{U}, \boldsymbol{U}^*) \le \delta_t$. Consider any $\ell \in \mathcal{J}_{good}$. Let $\boldsymbol{g}_k := \boldsymbol{U}^\top\boldsymbol{x}_k^* = (\boldsymbol{U}^\top\boldsymbol{U}^*)\boldsymbol{b}_k^*$.

If $\delta_t \le 0.02/\kappa^2$, and if $\widetilde{m} \gtrsim \max(\log q, \log n, r)$, for $\epsilon_1 < 0.1$ then,
w.p. at least, $1 - \exp(\log q + r - c\epsilon_1^2\widetilde{m})$

1) $\|(\boldsymbol{b}_k)_\ell - \boldsymbol{g}_k\| \le 1.7\epsilon_1\delta_t\|\boldsymbol{b}_k^*\|$
2) $\|(\boldsymbol{b}_k)_\ell\| \le 1.1\|(\boldsymbol{b}_\ell^*)_k\|$
3) $\|\boldsymbol{B}_\ell - \boldsymbol{G}\|_F \le 1.7\epsilon_1\delta_t\sigma_1^*$
4) $\|(\boldsymbol{x}_\ell)_k - \boldsymbol{x}_k^*\| \le 1.4\delta_t\|\boldsymbol{b}_k^*\|$
5) $\|\boldsymbol{X}_\ell - \boldsymbol{X}^*\|_F \le 1.4\delta_t\sigma_1^*$
6) $\sigma_r(\boldsymbol{B}_\ell) \ge 0.9\sigma_r^*$
7) $\sigma_{\max}(\boldsymbol{B}_\ell) \le 1.1\sigma_1^*$

(only the last two bounds require the upper bound on δ_t).

All the lemmas given below for analyzing the GD step use Lemma 7 in their proofs.

B. Lemmas for Proving Theorem 5.3 for $\tilde{L} = L$: GD Step Bounds

The main goal here is to bound $\boldsymbol{SD}_F(\boldsymbol{U}^+, \boldsymbol{U}^*)$, given that $\boldsymbol{SD}_F(\boldsymbol{U}, \boldsymbol{U}^*) \le \delta_t$. Here $\boldsymbol{U}^+$ is the subspace estimate at the next, $(t+1)$-th iteration. We will show that $\boldsymbol{SD}_F(\boldsymbol{U}^+, \boldsymbol{U}^*) \le (1-(\eta\sigma_1^{*2})\frac{c}{\kappa^2})\delta_t$. In our previous work [49], [50], we obtained this by bounding the deviation of the gradient, $\nabla f = \sum_{k\in[q]}\nabla f_k$ from its expected value, $\mathbb{E}[\nabla f] = m(\boldsymbol{X} - \boldsymbol{X}^*)\boldsymbol{B}^\top$ and then using this simple expression for the expected gradient to obtain the rest of our bounds. In particular notice that $\mathcal{P}_{\boldsymbol{U}^*,\perp}\mathbb{E}[\nabla f] = m\mathcal{P}_{\boldsymbol{U}^*,\perp}\boldsymbol{U}\boldsymbol{B}\boldsymbol{B}^\top$.

In this work, to use the same proof structure, we need a proxy for $\mathbb{E}[\nabla f]$. For this, we can use $\mathbb{E}[(\nabla f)_\ell]$ for any $\ell \in \mathcal{J}_{good}$. We let $\ell_1 \in \mathcal{J}_{good}$ be one such node. In what follows, we will use $\mathbb{E}[\nabla f_{\ell_1}] = m(\boldsymbol{X}_{\ell_1} - \boldsymbol{X}^*)\boldsymbol{B}_{\ell_1}^\top$ at various places.

Lemma 8: (algebra lemma) Let

$$\text{Err} = \nabla f_{GM} - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B})].$$

Recall that $\boldsymbol{U}^+ = QR(\boldsymbol{U} - (\eta/\widetilde{m})\nabla f_{GM})$. We have

$$\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}^+) \leq \frac{\|\boldsymbol{I}_r - \eta \boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top\| \boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}) + \frac{\eta}{\widetilde{m}}\|\mathrm{Err}\|_F}{1 - \frac{\eta}{\widetilde{m}}\|\mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B})]\| - \frac{\eta}{\widetilde{m}}\|\mathrm{Err}\|}$$

Proof: See Appendix B-D. □

Lemma 9: Assume $\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}) \leq \delta_t < \delta_0$.

1) If $\eta \leq 0.5/\sigma_1^{*2}$, $\lambda_{\min}(\boldsymbol{I}_r - \eta \boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top) = 1 - \eta\|\boldsymbol{B}_{\ell_1}\|^2 > 1 - \eta 1.3\sigma_1^{*2} > 0$ and so this matrix is p.s.d. and hence, $\|\boldsymbol{I}_r - \eta \boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top\| = \lambda_{\max}(\boldsymbol{I}_r - \eta \boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top) = 1 - \eta\sigma_{min}(\boldsymbol{B}_{\ell_1})^2 \leq 1 - \eta 0.8\sigma_r^{*2}$
2) For all $\ell \in \mathcal{J}_{good}$,
$$\mathbb{E}[\nabla f_\ell(\boldsymbol{U}, \boldsymbol{B}_\ell)] = \widetilde{m}(\boldsymbol{X}_\ell - \boldsymbol{X}^*)\boldsymbol{B}_\ell^\top$$
3) For all $\ell \in \mathcal{J}_{good}$,
$$\|\mathbb{E}[\nabla f_\ell(\boldsymbol{U}, \boldsymbol{B}_\ell)]\|_F \leq \widetilde{m} 1.6\delta_t \sigma_1^{*2}$$

Proof: The first item follows using the bounds on $\sigma_1^*(\boldsymbol{B}_\ell)$ and $\sigma_r^*(\boldsymbol{B}_\ell)$ from Lemma 7. Second item is immediate. Third item follows from item two and bounds on $\sigma_1^*(\boldsymbol{B}_\ell)$, $\|\boldsymbol{X}_\ell - \boldsymbol{X}^*\|_F$ given in Lemma 7. □

The next lemma is an easy consequence of Lemmas 2 and Lemma 6.

Lemma 10: Let $p_1 = \exp\left((n+r) - c\epsilon_1^2 \frac{\widetilde{m}q}{r\mu^2}\right) + 2\exp(\log q + r - c\epsilon_1^2\widetilde{m})$. If $\tau < 0.4$, then, w.p. at least $1 - Lp_1 - \exp(-L\psi(0.4 - \tau, p_1))$,

$$\|\mathrm{Err}\|_F \leq 14.12.5\widetilde{m}\sigma_{\max}^{*2}\epsilon_1\sqrt{r}\delta_t$$

Proof: See Appendix B-F. □

C. Proof of Theorem 5.3

The proof is an easy consequence of the above lemmas. Using the bounds from Lemma 10, 9 and the $\boldsymbol{SD}_F$ bound from Lemma 8, setting $\epsilon_1 = 0.3/175\sqrt{r}\kappa^2$, and using $\delta_t \leq \delta_0 = 0.1/\kappa^2$ in the denominator terms, we conclude the following: if in each iteration, $\widetilde{m}q \geq C_1\kappa^4\mu^2(n+r)r^2$, $\widetilde{m} > C_2\max(\log q, \log n)$, then, w.p. at least $1 - Lp_1 - \exp(-L\psi(0.4-\tau, p_1))$, where $p_1 = \exp\left((n+r) - c\frac{\widetilde{m}q}{r^2\kappa^4\mu^2}\right) + 2\exp(\log q + r - c\widetilde{m}/\kappa^4)$

$$\boldsymbol{SD}_F(\boldsymbol{U}^*, \boldsymbol{U}^+) \leq \frac{(1 - \frac{0.8\eta\sigma_1^{*2}}{\kappa^2})\delta_t + \frac{0.3\eta\sigma_1^{*2}}{\kappa^2}\delta_t}{1 - \frac{1.6\cdot 0.1\eta\sigma_1^{*2}}{\kappa^2} - \frac{0.1\cdot 0.3\eta\sigma_1^{*2}}{\kappa^2}}$$

$\leq (1 - (\eta\sigma_1^{*2})\frac{c}{\kappa^2})\delta_t := \delta_{t+1}$ Applying this bound at each t proves the theorem.

The numerical constants may have minor errors in various places.

D. Proof of Algebra Lemma, Lemma 8

Recall that $\mathrm{Err} = \nabla f_{GM} - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})]$. Let $\mathcal{P} := \boldsymbol{I} - \boldsymbol{U}^*\boldsymbol{U}^{*T}$.

GD step is given as $\hat{\boldsymbol{U}}^+ = \boldsymbol{U} - \frac{\eta}{\widetilde{m}}\nabla f_{GM}$.

Adding and subtracting $\mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})] = \widetilde{m}(\boldsymbol{X}_{\ell_1} - \boldsymbol{X}^*)\boldsymbol{B}_{\ell_1}^\top$, we get

$$\hat{\boldsymbol{U}}^+ = \boldsymbol{U} - \frac{\eta}{\widetilde{m}}\widetilde{m}(\boldsymbol{U}\boldsymbol{B}_{\ell_1} - \boldsymbol{X}^*)\boldsymbol{B}_{\ell_1}^\top - \frac{\eta}{\widetilde{m}}\mathrm{Err} \tag{14}$$

Multiplying both sides by $\mathcal{P} := \boldsymbol{I} - \boldsymbol{U}^*\boldsymbol{U}^{*\top}$,

$$\begin{aligned}\mathcal{P}\hat{\boldsymbol{U}}^+ &= \mathcal{P}\boldsymbol{U} - \eta\mathcal{P}\boldsymbol{U}\boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top - \frac{\eta}{\widetilde{m}}\mathcal{P}\mathrm{Err} \\ &= \mathcal{P}\boldsymbol{U}(\boldsymbol{I}_r - \eta\boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top) - \frac{\eta}{\widetilde{m}}\mathcal{P}\mathrm{Err}\end{aligned} \tag{15}$$

Taking Frobenius norm and using $\|M_1M_2\|_F \leq \|M_1\|_F\|M_2\|$ we get

$$\|\mathcal{P}\hat{\boldsymbol{U}}^+\|_F \leq \|\mathcal{P}\boldsymbol{U}\|_F\|I_r - \eta\boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top\| + \frac{\eta}{\widetilde{m}}\|\mathcal{P}\mathrm{Err}\|_F \tag{16}$$

Now $\hat{\boldsymbol{U}}^+ \overset{QR}{=} \boldsymbol{U}^+R^+$ and since $\|M_1M_2\|_F \leq \|M_1\|_F\|M_2\|$, this means that $\boldsymbol{SD}(\boldsymbol{U}^*, \boldsymbol{U}^+) \leq \|(I - \boldsymbol{U}^*\boldsymbol{U}^{*T})\hat{\boldsymbol{U}}^+\|_F\|(R^+)^{-1}\|$. Since $\|(R^+)^{-1}\| = 1/\sigma_{min}(R^+) = 1/\sigma_{min}(\hat{\boldsymbol{U}}^+)$,

$$\begin{aligned}\|(R^+)^{-1}\| &= \frac{1}{\sigma_{min}(\boldsymbol{U} - \frac{\eta}{\widetilde{m}}(\mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})] + \mathrm{Err}))} \\ &\leq \frac{1}{1 - \frac{\eta}{\widetilde{m}}\|\mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})]\| - \frac{\eta}{\widetilde{m}}\|\mathrm{Err}\|}\end{aligned}$$

Combining the last two bounds proves our result.

E. Bounding $\|\nabla f_\ell(U, B_\ell) - \mathbb{E}[\nabla f_{\ell_l}(U, B_{\ell_l})]\|_F$: Proof of Lemma 2

From the proof of [49, Lemma 3.5 item 1] we can write w.p. at least $1 - \exp\left((n+r) - c\epsilon_1^2\frac{\widetilde{m}q}{r\mu^2}\right) - 2\exp(\log q + r - c\epsilon_1^2\widetilde{m})$

$$\|\nabla f_\ell - \mathbb{E}[\nabla f_\ell]\|_F \leq 1.5\epsilon_1\sqrt{r}\delta_t\widetilde{m}\sigma_{\max}^{*2} \tag{17}$$

Using (17) and Lemma 9, item 2,

$$\begin{aligned}&\|\nabla f_\ell - \mathbb{E}[\nabla f_{\ell_1}]\|_F \leq \\ &\|\nabla f_\ell - \mathbb{E}[\nabla f_\ell]\|_F + \|\mathbb{E}[\nabla f_\ell] - \mathbb{E}[\nabla f_{\ell_1}]\|_F \leq \\ &1.5\epsilon_1\sqrt{r}\delta_t\widetilde{m}\sigma_{\max}^{*2} \\ &+ \|\widetilde{m}(\boldsymbol{X}_\ell - \boldsymbol{X}^*)\boldsymbol{B}_\ell^\top - \widetilde{m}(\boldsymbol{X}_{\ell_1} - \boldsymbol{X}^*)\boldsymbol{B}_{\ell_1}^\top\|_F\end{aligned} \tag{18}$$

Using the bounds from Lemma 7,

$$\begin{aligned}&\|\widetilde{m}(\boldsymbol{X}_\ell - \boldsymbol{X}^*)\boldsymbol{B}_\ell^\top - \widetilde{m}(\boldsymbol{X}_{\ell_1} - \boldsymbol{X}^*)\boldsymbol{B}_{\ell_1}^\top\|_F \\ &= \widetilde{m}\|\boldsymbol{U}(\boldsymbol{B}_\ell\boldsymbol{B}_\ell^\top - \boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top) - \boldsymbol{X}^*(\boldsymbol{B}_\ell^\top - \boldsymbol{B}_{\ell_1}^\top)\|_F \\ &= \widetilde{m}\|\boldsymbol{U}(\boldsymbol{B}_\ell\boldsymbol{B}_\ell^\top - \boldsymbol{B}_{\ell_1}\boldsymbol{B}_{\ell_1}^\top \pm \boldsymbol{B}_\ell\boldsymbol{B}_{\ell_1}^\top) - \boldsymbol{X}^*(\boldsymbol{B}_\ell - \boldsymbol{B}_{\ell_1})^\top\|_F \\ &= \widetilde{m}\Big\|\boldsymbol{U}\boldsymbol{B}_\ell(\boldsymbol{B}_\ell^\top - \boldsymbol{B}_{\ell_1}^\top) - \boldsymbol{U}(\boldsymbol{B}_{\ell_1} - \boldsymbol{B}_\ell)\boldsymbol{B}_{\ell_1}^\top \\ &\qquad - \boldsymbol{X}^*(\boldsymbol{B}_\ell^\top - \boldsymbol{B}_{\ell_1}^\top)\Big\|_F \\ &\leq \widetilde{m}(1.1\sigma_1^* + 1.1\sigma_1^* + \sigma_1^*)\|\boldsymbol{B}_\ell - \boldsymbol{B}_{\ell_1}\|_F \\ &= \widetilde{m}3.2\sigma_1^*\|\boldsymbol{B}_\ell - \boldsymbol{B}_{\ell_1} \pm \boldsymbol{G}\|_F \\ &\leq \widetilde{m}3.2\sigma_1^*(\|\boldsymbol{B}_\ell - \boldsymbol{G}\|_F + \|\boldsymbol{B}_{\ell_1} - \boldsymbol{G}\|_F) \leq \widetilde{m}11\sigma_1^{*2}\epsilon_1\sqrt{r}\delta_t\end{aligned}$$

Using this in (18)

$$\begin{aligned}\|\nabla f_\ell - \mathbb{E}[\nabla f_{\ell_1}]\|_F &\leq 1.5\epsilon_1\sqrt{r}\delta_t\widetilde{m}\sigma_1^{*2} + \widetilde{m}11\sigma_1^{*2}\epsilon_1\sqrt{r}\delta_t \\ &\leq \widetilde{m}12.5\sigma_1^{*2}\epsilon_1\sqrt{r}\delta_t\end{aligned} \tag{19}$$

w.p. at least $1 - \exp\left((n+r) - c\epsilon_1^2\frac{\widetilde{m}q}{r\mu^2}\right) - 2\exp(\log q + r - c\epsilon_1^2\widetilde{m})$.

F. Bounding Err: Proof of Lemma 10

Recall that $\text{Err} = \nabla f_{GM} - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})]$. This bound follows from the Lemma 6 and Lemma 2. We apply Lemma 6 with $z_\ell \equiv \nabla f_\ell$ and $\tilde{z} \equiv \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})]$, $\alpha = 0.4$, $\tau < 0.4$, $\epsilon \equiv 7.8\epsilon_1 = \epsilon_{GM}$ and ω_{GM} set to a constant C times an upper bound on $\|\mathbb{E}[\nabla f_{\ell_1}]\|_F$. From Lemma 9, $\|\mathbb{E}[\nabla f_{\ell_1}]\|_F \le 2\widetilde{m}\delta_t{\sigma_1^*}^2 \le 2\widetilde{m}\delta_0{\sigma_1^*}^2$. The Theorem needs $\delta_0 = c/\kappa^2$. Thus, we can set $\omega_{GM} = C\widetilde{m}{\sigma_r^*}^2$.

To apply Lemma 6, we need a high probability bound on $\max_{\ell\in\mathcal{J}_{good}} \| \nabla f_\ell - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})]\|_F$.

By Lemma 2 and union bound and using $|\mathcal{J}_{good}| = (1-\tau)L \le L$, we can show that

$$\max_{\ell\in\mathcal{J}_{good}} \|\nabla f_\ell(\boldsymbol{U}, \boldsymbol{B}_\ell) - \mathbb{E}[\nabla f_{\ell_1}(\boldsymbol{U}, \boldsymbol{B}_{\ell_1})]\|_F \tag{20}$$

$$\le \widetilde{m} 12.5 {\sigma^*_{\max}}^2 \epsilon_1 \sqrt{r}\delta_t \tag{21}$$

w.p. at least $1 - L\Big(\exp\Big((n+r) - c\epsilon_1^2\frac{\widetilde{m}q}{r\mu^2}\Big) + 2\exp(\log q + r - c\epsilon_1^2\widetilde{m})\Big) := 1 - p_1$.

Thus, using Lemma 6 w.p. at least $1 - Lp_1 - \exp(-L\psi(0.4-\tau, p_1))$,

$$\|\text{Err}\|_F \le 14\widetilde{m}12.5{\sigma^*_{\max}}^2\epsilon_1\sqrt{r}\delta_t$$

APPENDIX C
ONE STEP ANALYSIS OF RESPOWMETH

If we want to analyze ResPowMeth directly, we need to bound $\boldsymbol{SD}(\hat{\boldsymbol{U}}, \boldsymbol{U}^*)$ at each iteration. Consider its first iteration.

Let $\mathcal{P}_{\boldsymbol{U}^*,\perp} = \boldsymbol{I} - \boldsymbol{U}^*\boldsymbol{U}^{*\top}$, $GM = GM\{\boldsymbol{\Phi}_\ell \boldsymbol{U}_{rand}\}_{\ell=1}^L$. Then,

$$\boldsymbol{SD}_F(\boldsymbol{U}^*, \hat{\boldsymbol{U}}) \le \|\mathcal{P}_{\boldsymbol{U}^*,\perp} GM\|_F \|(R^+)^{-1}\| \le \frac{\|\mathcal{P}_{\boldsymbol{U}^*,\perp} GM\|_F}{\sigma_{min}(GM)}$$

Here $GM \overset{QR}{=} \hat{\boldsymbol{U}}R^+$. This follows since $\|(R^+)^{-1}\| = 1/\sigma_{min}(R^+) = 1/\sigma_{min}(GM)$.

To bound both numerator and denominator, we use the fact that GM is an approximation of $\boldsymbol{\Phi}^*\boldsymbol{U}_{rand}$.

Suppose that

$$\|\boldsymbol{\Phi}_\ell - \boldsymbol{\Phi}^*\| \le b_0.$$

Using [56, Theorem 4.4.5], $\|\boldsymbol{U}_{rand}\| \le 1.1(\sqrt{n} + \sqrt{r})$, and so,

$$\|\boldsymbol{\Phi}_\ell \boldsymbol{U}_{rand} - \boldsymbol{\Phi}^*\boldsymbol{U}_{rand}\| \le b_0\|\boldsymbol{U}_{rand}\| \le 2.2 b_0\sqrt{n}$$

where we used $r \le n$. Using this and applying Lemma 6 with $\epsilon_{GM} = b_0$, we have w.p. at least $1 - c_0 - Lp - \exp(-L\psi(0.4-\tau, p))$

$$\|GM - \boldsymbol{\Phi}^*\boldsymbol{U}_{rand}\| \le 31\ b_0\sqrt{n}$$

Then,

$$\begin{aligned}
&\|\mathcal{P}_{\boldsymbol{U}^*,\perp} GM\|_F \\
&= \|\mathcal{P}_{\boldsymbol{U}^*,\perp}(GM - \boldsymbol{\Phi}^*\boldsymbol{U}_{rand} + \boldsymbol{\Phi}^*\boldsymbol{U}_{rand})\|_F \\
&= \|\mathcal{P}_{\boldsymbol{U}^*,\perp}(GM - \boldsymbol{\Phi}^*\boldsymbol{U}_{rand})\|_F + \|\mathcal{P}_{\boldsymbol{U}^*,\perp}\boldsymbol{\Phi}^*\boldsymbol{U}_{rand})\|_F \\
&\le \|GM - \boldsymbol{\Phi}^*\boldsymbol{U}_{rand}\|_F + \sigma^*_{r+1}\|\boldsymbol{U}_{rand}\|\sqrt{r} \\
&= 31 b_0\sqrt{n} + \sigma^*_{r+1}\cdot 2.2\sqrt{n}\cdot\sqrt{r}
\end{aligned} \tag{22}$$

where we used $\|\mathcal{P}_{\boldsymbol{U}^*,\perp}\boldsymbol{\Phi}^*\boldsymbol{U}_{rand})\|_F \le \|\mathcal{P}_{\boldsymbol{U}^*,\perp}\boldsymbol{\Phi}^*\| \|\boldsymbol{U}_{rand})\|_F \le \sigma^*_{r+1}\|\boldsymbol{U}_{rand}\|\sqrt{r}$. Also,

$$\begin{aligned}
&\sigma_{min}(GM) \\
&\ge \sigma_{min}(\boldsymbol{\Phi}^*\boldsymbol{U}_{rand}) - \|\boldsymbol{\Phi}^*\boldsymbol{U}_{rand} - GM\| \\
&\ge \sigma_{min}(\boldsymbol{\Phi}^*\boldsymbol{U}_{rand}) - 31 b_0\sqrt{n} \\
&\ge \sigma_{min}(\mathcal{P}_{\boldsymbol{U}^*}\boldsymbol{\Phi}^*\boldsymbol{U}_{rand}) - \|\mathcal{P}_{\boldsymbol{U}^*,\perp}\boldsymbol{\Phi}^*\boldsymbol{U}_{rand}\| - 31 b_0\sqrt{n}
\end{aligned} \tag{23}$$

$$\ge \sigma_{min}(\mathcal{P}_{\boldsymbol{U}^*}\boldsymbol{\Phi}^*\boldsymbol{U}_{rand}) - 2.2\sigma^*_{r+1}\sqrt{n} - 31 b_0\sqrt{n} \tag{24}$$

where we used Weyl's inequality and $\boldsymbol{\Phi}^*\boldsymbol{U}_{rand} = \mathcal{P}_{\boldsymbol{U}^*}\boldsymbol{\Phi}^*\boldsymbol{U}_{rand} + \mathcal{P}_{\boldsymbol{U}^*,\perp}\boldsymbol{\Phi}^*\boldsymbol{U}_{rand}$ Finally,

$$\begin{aligned}
\sigma_{min}(\mathcal{P}_{\boldsymbol{U}^*}\boldsymbol{\Phi}^*\boldsymbol{U}_{rand}) &= \sigma_{min}(\boldsymbol{U}^*\boldsymbol{\Sigma}\boldsymbol{U}^{*\top}\boldsymbol{U}_{rand}) \\
&\ge \sigma_{\min}(\boldsymbol{U}^*)\sigma^*_r\sigma_{min}\left(\boldsymbol{U}^{*\top}\boldsymbol{U}_{rand}\right)
\end{aligned} \tag{25}$$

We bound $\sigma_{min}\left(\boldsymbol{U}^{*\top}\boldsymbol{U}_{rand}\right)$ using [67, Theorem 1.1] which helps bound the minimum singular value of square matrices with i.i.d. zero-mean sub-Gaussian entries. i, j-th entry of $\boldsymbol{U}^{*\top}\boldsymbol{U}_{rand}$ is the inner product of i-th column of $\boldsymbol{U}^*$ and $j-$th column of $\boldsymbol{U}_{rand}$. $\boldsymbol{U}^*$ has orthonormal columns and hence each entry of $\boldsymbol{U}^{*\top}\boldsymbol{U}_{rand}$ is mean-zero, unit variance Gaussian r.v. Thus, by [67, Theorem 1.1], w.p., at least $1 - (C\epsilon) - \exp^{-cr}$

$$\begin{aligned}
\sigma_{min}(\boldsymbol{U}^{*\top}\boldsymbol{U}_{rand}) &\ge \epsilon(\sqrt{r} - \sqrt{r-1}) \\
&= \epsilon\sqrt{r}\left(1 - \sqrt{1 - \frac{1}{r}}\right) \\
&\ge \epsilon\sqrt{r}\left(1 - \left(1 - \frac{1}{2r}\right)\right) \\
&= \epsilon\frac{1}{2\sqrt{r}}
\end{aligned}$$

In the above, we used Bernoulli inequality $(1+x)^n \le 1 + nx$, where $0 \le n \le 1$, $x \ge -1$ for $\sqrt{1-\frac{1}{r}}$. Use $\epsilon = 0.1$.

Thus, w.p. at least $1 - 0.1 - \exp^{-cr}$,

$$\sigma_{min}(GM) \ge \sigma^*_r 0.1\frac{1}{2\sqrt{r}} - 2.2\sigma^*_{r+1}\sqrt{n} - 31 b_0\sqrt{n}$$

Together this implies

$$\boldsymbol{SD}_F(\boldsymbol{U}^*, \hat{\boldsymbol{U}}) \le \frac{31 b_0\sqrt{n} + \sigma^*_{r+1}\cdot 2.2\sqrt{n}\cdot\sqrt{r}}{\sigma^*_r 0.1\frac{1}{2\sqrt{r}} - 2.2\sigma^*_{r+1}\sqrt{n} - 31 b_0\sqrt{n}} \tag{26}$$

$$\le \frac{62 b_0\sqrt{nr} + 4.4\sigma^*_{r+1}\sqrt{n}\cdot r}{\sigma^*_r 0.1 - 4.4\sigma^*_{r+1}\sqrt{nr} - 62 b_0\sqrt{nr}} \tag{27}$$

To get this bound below ϵ_1, we need $b_0 \le c\epsilon_1/\sqrt{nr}$ and we need $\sigma^*_{r+1} < c/(\sqrt{n}r)$. Thus even to get $\epsilon_1 = 0.99$ (any value less than one), we need b_0 to be of order $1/\sqrt{nr}$.

APPENDIX D
GEOMETRIC MEDIAN COMPUTATION ALGORITHMS

The geometric median cannot be computed exactly. We describe below two algorithms to compute it. The first is the approach developed in Cohen et al [60]. This comes with a near-linear computational complexity bound. However, as we briefly explain below this is very complicated to implement and needs too many parameters. No numerical simulation results have been reported using this approach even in [60] itself and not in works that cite it either (to our best knowledge).

The practically used GM approach is Weiszfeld's algorithm [59], which is a form of iteratively re-weighted least squares algorithms. It is simple to implement and works well in practice. However it either comes with an asymptotic guarantee, or with a finite time guarantee for which the bound on the required number of iterations is not easy to interpret. This bound depends upon the chosen initialization for the algorithm. Because of this, we cannot provide an easily interpretable bound on its computational complexity.

Algorithm 6 `AccurateMedian`(ϵ_{GM})

Input: points $z_1, \ldots, z_L \in \mathbb{R}^d$ **Input**: desired accuracy $\epsilon_{GM} \in (0,1)$

Compute a 2-approximate geometric median and use it to center
Compute $x^{(0)} := \frac{1}{L}\sum_{i\in[L]} z_i$ and $\widetilde{f}_* := f(x^{(0)})$
{Here $f(x) = \sum_{i\in[L]} \|x - z_i\|_2$}
Let $t_i = \frac{1}{400\widetilde{f}_*}(1+\frac{1}{600})^{i-1}$, $\tilde{\epsilon}_* = \frac{1}{3}\epsilon_{GM}$, and $\tilde{t}_* = \frac{2L}{\tilde{\epsilon}_*\cdot\widetilde{f}_*}$
Let $\epsilon_v = \frac{1}{8}(\frac{\tilde{\epsilon}_*}{7L})^2$ and let $\epsilon_c = (\frac{\epsilon_v}{36})^{\frac{3}{2}}$
$x^{(1)} = \texttt{LineSearch}(x^{(0)}, t_1, t_1, 0, \epsilon_c)$
Iteratively improve quality of approximation
Let $T_{GM} = \max_{i\in\mathbb{Z}} t_i \le \tilde{t}_*$
for $i \in [1, T_{GM}]$ **do**
  Compute ϵ_v-approximate minimum eigenvalue and eigenvector of $\nabla^2 f_{t_i}(x^{(i)})$
  $(\lambda^{(i)}, u^{(i)}) = \texttt{ApproxMinEig}(x^{(i)}, t_i, \epsilon_v)$
  Line search to find $x^{(i+1)}$ such that $\|x^{(i+1)} - x_{t_{i+1}}\|_2 \le \frac{\epsilon_c}{t_{i+1}}$
  $x^{(i+1)} = \texttt{LineSearch}(x^{(i)}, t_i, t_{i+1}, u^{(i)}, \epsilon_c)$
end for
Output: ϵ_{GM}-approximate geometric median $x^{(T_{GM}+1)}$.

A. Cohen Et Al [60]'s Algorithm: Nearly Linear Time GM

The function `ApproxMinEig` in Algorithm 6 calculates an approximation of the minimum eigenvector of $\nabla^2 f_t(x)$. This approximation is obtained using the well-known power method, which converges rapidly on matrices with a large eigenvalue gap. By leveraging this property, we can obtain a concise approximation of $\nabla^2 f_t(x)$. The running time of `ApproxMinEig` is $\mathcal{O}\left(Ld\log\left(\frac{L}{\epsilon_{GM}}\right)\right)$. This time complexity indicates that the algorithm's execution time grows linearly with L and d and logarithmically with L/ϵ_{GM}. The function `LineSearch` in Algorithm 6 performs a line search on the function $g_{t,y,v}(\alpha)$, as defined in Equation 28. The line search aims to find the minimum value of $g_{t,y,v}(\alpha)$, subject to the constraint $|\mathbf{x}-(\mathbf{y}+\alpha\mathbf{v})|_2 \le \frac{1}{49t}$, where $\mathbf{x}$ is the variable being optimized.

$$g_{t,y,v}(\alpha) = \min_{\|x-(y+\alpha v)\|_2 \le \frac{1}{49t}} f_t(x) \tag{28}$$

To evaluate $g_{t,y,v}(\alpha)$ approximately, an appropriate centering procedure is utilized. This procedure allows for an efficient estimation of the function's value. The running time of `LineSearch` is $\mathcal{O}\left(Ld\log^2\left(\frac{L}{\epsilon_{GM}}\right)\right)$. The time complexity indicates that the algorithm's execution time grows linearly with L and d, while the logarithmic term accounts for the influence of $\frac{L}{\epsilon_{GM}}$ on the running time.

B. Practical Algorithm: Weiszfeld's Method

Weiszfeld's algorithm, Algorithm 7, provides a simpler approach for approximating the Geometric Median (GM). It is easier to implement compared to Algorithm 6. It is an iteratively reweighted least squares algorithm. It iteratively refines the estimate by giving higher weights to points that are closer to the current estimate, effectively pulling the estimate towards the dense regions of the point set. The process continues until a desired level of approximation is achieved, often determined by a tolerance parameter, ϵ_{GM}. While the exact number of iterations needed cannot be determined theoretically (as we will see from its guarantees below), the algorithm typically converges reasonably quickly in practice.

We provide here the two known guarantees for this algorithm.

Theorem D.4: [Corollary 7.1 [59]] Suppose that there is no optimal $\boldsymbol{z} \in \mathcal{A} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_L\}$ such that it minimizes $\sum_{\ell=1}^{L}\|\boldsymbol{z} - \boldsymbol{z}_\ell\|$. Let $\{z_t\}_{t\ge 0}$ be the sequence generated by Weiszfeld's Algorithm 7 with z_0 as given in the initialization of Algorithm 7. Then, for any $t \ge 0$, we have $\boldsymbol{z}_t \notin \mathcal{A}$ and $\boldsymbol{z}_t \to \boldsymbol{z}^*$ as $t \to \infty$. Here z^* is the true GM.

Theorem D.5: [Theorem 8.2 [59]] Let $\{z_t\}_{t\ge 0}$ be the sequence generated by Weiszfeld's Algorithm 7 with z_0 as given in the initialization of Algorithm 7. Then, for any $t \ge 0$, we have

$$f(z_t) - f^* \le \frac{M}{2t}\|z_0 - z^*\|^2$$

where $M = \frac{2\mathcal{L}(z_p)L^2}{(\|R_p\|-1)^2}$. Here z^* is the true GM.

The first result above is asymptotic. The second one, Theorem D.5, gives convergence rate of $\mathcal{O}(M/t)$ where M is as defined in the theorem. It is not clear how to upper bound M only in terms of the model parameters (d, L or z^*). Consequently, the rate of convergence is not clear. Moreover, the expression for z_0 (initialization) is too complicated and thus it is not clear how to bound $\|z_0 - z^*\|^2$. Consequently, one cannot provide an expression for the iteration complexity that depends only on the model parameters.

C. Proof of Lemma 3

$\mathcal{J}_{good} = \{\ell : \|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \le \epsilon\|\tilde{\boldsymbol{z}}\|\}$ and define $z^* := GM(z_1, \ldots, z_L)$ as exact Geometric median.

Algorithm 7 Weiszfeld's Algorithm

Input $\mathcal{A} = \{z_1, z_2, \ldots, z_L\}$
Parameters T, ϵ_{GM}
Output z_{GM}
Initialization
$z_0 = z_p + t_p d_p$, where
$p \in \arg\min\{f(z_i) : 1 \le i \le L\}$, $f(z) = \sum_{i=1}^{L} \|z - z_i\|$
and $d_p = \frac{R_p}{\|R_p\|}$, $t_p = \frac{\|R_p\|-1}{\mathcal{L}(z_p)}$

$$\mathcal{L}(z) = \begin{cases} \sum_{i=1}^{L} \frac{1}{\|z-z_i\|} & \text{if} \quad z \notin \mathcal{A} \\ \sum_{i=1, i\neq j}^{L} \frac{1}{\|z_j - z_i\|} & \text{if} \quad z = z_j \quad (1 \le j \le L) \end{cases}$$

$$R_j = \sum_{i=1, i\neq j}^{L} \frac{z_j - z_i}{\|z_i - z_j\|}$$

Iterative step

$$z_{t+1} = \left(\sum_{i=1}^{L} \frac{z_i}{\|z_i - z_t\|}\right) \Big/ \left(\sum_{i=1}^{L} \frac{1}{\|z_i - z_t\|}\right)$$

Terminating Condition

1) $t > T$ Upper bound on number of iterations
2) $\|z_{t+1} - z_t\|_2 < \epsilon_{GM}$

For $\boldsymbol{z}_\ell$ with $\ell \in \mathcal{J}_{good}$, we have

$$\begin{aligned} \|\boldsymbol{z}_{GM} - \boldsymbol{z}_\ell\| &= \|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}} + \tilde{\boldsymbol{z}} - \boldsymbol{z}_\ell\| \\ &\ge \|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| - \|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \\ &\ge \|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| - 2\epsilon\|\tilde{\boldsymbol{z}}\| + \|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \end{aligned} \tag{29}$$

Moreover, by triangle inequality for $\boldsymbol{z}_\ell \notin \mathcal{J}_{good}$, we have

$$\|\boldsymbol{z}_{GM} - \boldsymbol{z}_\ell\| \ge \|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| - \|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| \tag{30}$$

Summing (29), (30) we get

$$\begin{aligned} \sum_{\ell=1}^{L} \|\boldsymbol{z}_{GM} - \boldsymbol{z}_\ell\| \ge \sum_{\ell=1}^{L} \|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \\ +(2|\mathcal{J}_{good}| - L)\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| - 2|\mathcal{J}_{good}|\epsilon\|\tilde{\boldsymbol{z}}\| \end{aligned}$$

By definition of $\boldsymbol{z}_{GM}$ (approximate GM), $\sum_{\ell=1}^{L} \|\boldsymbol{z}_{GM} - \boldsymbol{z}_\ell\| \le (1+\epsilon_{GM}) \sum_{\ell=1}^{L} \|\boldsymbol{z}^* - \boldsymbol{z}_\ell\|$. Hence,

$$\begin{aligned} \sum_{\ell=1}^{L} \|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| + (2|\mathcal{J}_{good}| - L)\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| - 2|\mathcal{J}_{good}|\epsilon\|\tilde{\boldsymbol{z}}\| \\ \le (1+\epsilon_{GM}) \sum_{\ell=1}^{L} \|\boldsymbol{z}^* - \boldsymbol{z}_\ell\| \end{aligned}$$

Since $\boldsymbol{z}^*$ is the minimizer of $\min_{\boldsymbol{z}\in\Re^n} \sum_{\ell=1}^{L} \|\boldsymbol{z} - \boldsymbol{z}_\ell\|$, so

$$\sum_{\ell=1}^{L} \|\boldsymbol{z}^* - \boldsymbol{z}_\ell\| \le \sum_{\ell=1}^{L} \|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\|.$$

Using this to lower bound the first term on the LHS of above,

$$\begin{aligned} &\sum_{\ell=1}^{L} \|\boldsymbol{z}^* - \boldsymbol{z}_\ell\| + (2|\mathcal{J}_{good}| - L)\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| - 2|\mathcal{J}_{good}|\epsilon\|\tilde{\boldsymbol{z}}\| \\ &\le (1+\epsilon_{GM}) \sum_{\ell=1}^{L} \|\boldsymbol{z}^* - \boldsymbol{z}_\ell\| \end{aligned}$$

Arranging the terms and using the fact $|\mathcal{J}_{good}| \ge (1-\alpha)L$ we get

$$\begin{aligned} &\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| \\ &\le \frac{2|\mathcal{J}_{good}|\epsilon\|\tilde{\boldsymbol{z}}\|}{2|\mathcal{J}_{good}| - L} + \epsilon_{GM} \frac{\sum_{\ell=1}^{L} \|\boldsymbol{z}^* - \boldsymbol{z}_\ell\|}{2|\mathcal{J}_{good}| - L} \\ &\le \frac{2(1-\alpha)\epsilon\|\tilde{\boldsymbol{z}}\|}{1-2\alpha} + \epsilon_{GM} \frac{\max_{\ell\in[L]} \|\boldsymbol{z}_\ell\|}{1-2\alpha} \end{aligned}$$

Using Claim 1 (with constant probability $1 - c_{approxGM}$ Algorithm 6 obtains $(1+\epsilon_{GM})-$approximate geometric median $\boldsymbol{z}_{GM}$ in order $T_{GM} = C \log\left(\frac{L}{\epsilon_{GM}}\right)$) implies that with probability $1 - c_{approxGM}$

$$\begin{aligned} \|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| &\le C_\alpha \epsilon \|\tilde{\boldsymbol{z}}\| + \epsilon_{GM} \frac{\sum_{\ell=1}^{L} \|\boldsymbol{z}^* - \boldsymbol{z}_\ell\|}{(1-2\alpha)L} \\ &\le C_\alpha \epsilon \|\tilde{\boldsymbol{z}}\| + \epsilon_{GM} \frac{\max_{\ell\in[L]} \|\boldsymbol{z}_\ell\|}{1-2\alpha} \end{aligned}$$

where $C_\alpha := \frac{2(1-\alpha)}{1-2\alpha}$.

D. Proof of Lemma 4

Given

$$\Pr\{\|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \le \epsilon\|\tilde{\boldsymbol{z}}\|\} \ge 1 - p$$

then

$$\begin{aligned} &\Pr\left\{\sum_{\ell=1}^{L} \mathbb{1}_{\{\|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \le \epsilon\|\tilde{\boldsymbol{z}}\|\}} \ge L(1-\alpha) + L_{byz}\right\} \\ &\ge \Pr\{T \ge L(1-\alpha) + L_{byz}\} \end{aligned}$$

where $T \sim$ Binomial$(L, 1-p)$ (First-order stochastic domination)

By Chernoff's bound for binomial distributions, the following holds:

$$\Pr\{T \ge L(1-\alpha) + L_{byz}\} \ge 1 - \exp(-L\psi(\alpha - \tau, p))$$

where $\tau = \frac{L_{byz}}{L}$

This then implies w.p. at least $1 - \exp(-L\psi(\alpha-\tau, p))$,

$$\sum_{\ell=1}^{L} \mathbb{1}_{\{\|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \le \epsilon\|\tilde{\boldsymbol{z}}\|\}} \ge L(1-\alpha) + L_{byz} \ge L(1-\alpha)$$

where $\alpha \in (\tau, 1/2)$. Using Lemma 3

$$\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| \le C_\alpha \epsilon \|\tilde{\boldsymbol{z}}\| + \epsilon_{GM} \frac{\max_{1\le\ell\le L} \|\boldsymbol{z}_\ell\|}{1-2\alpha}$$

w.p. at least $1 - c_{approxGM} - \exp(-L\psi(\alpha-\tau, p))$. Fixing $\alpha = 0.4$ we get the result.

E. Proof of Corollary 6

$\mathcal{J}_{good}$ denotes the set of good node (nodes whose estimates $\boldsymbol{z}_\ell$ satisfy $\|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \leq \epsilon\|\tilde{\boldsymbol{z}}\|$) with the stated probability. First we need to show that, with high probability, none of the entries of $\mathcal{J}_{good}$ are thresholded out. Using given condition $\Pr\{\|\boldsymbol{z}_\ell - \tilde{\boldsymbol{z}}\| \leq \epsilon\|\tilde{\boldsymbol{z}}\|\} \geq 1-p$ and union bound, we conclude that, w.p. at least $1-(1-\tau)Lp$, $\max_{\ell\in\mathcal{J}_{good}} \|\boldsymbol{z}_\ell\| \leq (1+\epsilon)\|\tilde{\boldsymbol{z}}\| = \omega_{GM}$. This means that, with this probability, none of the $\mathcal{J}_{good}$ elements are thresholded out.

For the set $\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_L\} \setminus \{\boldsymbol{z}_\ell : \|\boldsymbol{z}_\ell\| > (1+\epsilon)\|\tilde{\boldsymbol{z}}\|\}$ we apply Lemma 4. Since $|\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_L\} \setminus \{\boldsymbol{z}_\ell : \|\boldsymbol{z}_\ell\| > (1+\epsilon)\|\tilde{\boldsymbol{z}}\|\}| = L' \leq L$, $(1-\tau)L \leq L' \leq L$ implies $\tau' \leq \tau < 0.4$ hence condition of Lemma 4 is satisfied using Lemma 4 w.p. at least $1 - c_{approxGM} - \exp(-L'\psi(0.4-\tau',p)) - (1-\tau)Lp \geq 1 - c_{approxGM} - \exp(-L\psi(0.4-\tau,p)) - Lp$,

$$\|\boldsymbol{z}_{GM} - \tilde{\boldsymbol{z}}\| \leq 6\epsilon\|\tilde{\boldsymbol{z}}\| + 5\epsilon_{GM}(1+\epsilon)\|\tilde{\boldsymbol{z}}\| < 14\max(\epsilon,\epsilon_{GM})\|\tilde{\boldsymbol{z}}\|$$

REFERENCES

[1] A. P. Singh and N. Vaswani, "Byzantine-resilient federated principal subspace estimation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2024, pp. 2514–2519.

[2] A. P. Singh and N. Vaswani, "Byzantine resilient and fast federated few-shot learning," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 1–11.

[3] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[4] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3521–3530.

[5] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "DRACO: Byzantine-resilient distributed training via redundant gradients," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 903–912.

[6] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[7] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–11.

[8] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[9] S. G. Lingala, Y. Hu, E. DiBella, and M. Jacob, "Accelerated dynamic MRI exploiting sparsity and low-rank structure: K-t SLR," *IEEE Trans. Med. Imag.*, vol. 30, no. 5, pp. 1042–1054, May 2011.

[10] S. Babu, S. G. Lingala, and N. Vaswani, "Fast low rank compressive sensing for accelerated dynamic MRI," *IEEE Trans. Comput. Imag.*, vol. 9, pp. 409–424, 2023.

[11] S. S. Du, W. Hu, S. M. Kakade, J. D. Lee, and Q. Lei, "Few-shot learning via learning the representation, provably," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=pW2Q2xLwIMD

[12] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 2089–2099.

[13] H. Qi and S. M. Hughes, "Invariance of principal components under low-dimensional random projection of the data," in *Proc. 19th IEEE Int. Conf. Image Process.*, Sep. 2012, pp. 937–940.

[14] F. P. Anaraki and S. Hughes, "Memory and computation efficient PCA via very sparse random projections," in *Proc. Intl. Conf. Mach. Learn. (ICML)*, 2014, pp. 1341–1349.

[15] R. S. Srinivasa, K. Lee, M. Junge, and J. Romberg, "Decentralized sketching of low rank matrices," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 10101–10110.

[16] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.

[17] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6893–6901.

[18] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," 2020, *arXiv:2012.13995*.

[19] Z. Allen-Zhu, F. Ebrahimian, J. Li, and D. Alistarh, "Byzantine-resilient non-convex stochastic gradient descent," 2020, *arXiv:2012.14368*.

[20] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to Byzantine attacks," *IEEE Trans. Signal Process.*, vol. 68, pp. 4583–4596, 2020.

[21] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.

[22] A. Acharya, A. Hashemi, P. Jain, S. Sanghavi, I. S. Dhillon, and U. Topcu, "Robust training in high dimensions via block coordinate geometric median descent," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2022, pp. 11145–11168.

[23] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," 2019, *arXiv:1912.13445*.

[24] D. Data and S. Diggavi, "Byzantine-resilient SGD in high dimensions on heterogeneous data," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 2310–2315.

[25] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1544–1551.

[26] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," 2019, *arXiv:1906.06629*.

[27] J. Regatti, H. Chen, and A. Gupta, "Byzantine resilience with reputation scores," in *Proc. 58th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2022, pp. 1–8.

[28] S. Lu, R. Li, X. Chen, and Y. Ma, "Defense against local model poisoning attacks to Byzantine-robust federated learning," *Frontiers Comput. Sci.*, vol. 16, no. 6, Dec. 2022, Art. no. 166337.

[29] X. Cao and L. Lai, "Distributed gradient descent algorithm robust to an arbitrary number of Byzantine attackers," *IEEE Trans. Signal Process.*, vol. 67, no. 22, pp. 5850–5864, Nov. 2019.

[30] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, pp. 1–37, 2011.

[31] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM J. Optim.*, vol. 21, no. 2, pp. 572–596, Apr. 2011.

[32] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust PCA via gradient descent," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2016, pp. 1–9.

[33] P. Narayanamurthy and N. Vaswani, "Provable dynamic robust PCA or robust subspace tracking," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1547–1577, Mar. 2019.

[34] P. Narayanamurthy and N. Vaswani, "Fast robust subspace tracking via PCA in sparse data-dependent noise," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 3, pp. 723–744, Nov. 2020.

[35] T. Zhang and G. Lerman, "A novel M-estimator for robust PCA," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 749–808, 2014.

[36] X. Li, Z. Zhu, A. M.-C. So, and R. Vidal, "Nonconvex robust low-rank matrix recovery," *SIAM J. Optim.*, vol. 30, no. 1, pp. 660–686, Jan. 2020.

[37] P. Narayanamurthy, N. Vaswani, and A. Ramamoorthy, "Federated over-air subspace tracking from incomplete and corrupted data," *IEEE Trans. Signal Process.*, vol. 70, pp. 3906–3920, 2022.

[38] L. T. Thanh, A. M. Rekavandi, A.-K. Seghouane, and K. Abed-Meraim, "Robust subspace tracking with contamination mitigation via α-divergence," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[39] I. Diakonikolas, G. Kamath, D. M. Kane, J. Li, A. Moitra, and A. Stewart, "Being robust (in high dimensions) can be practical," in *Proc. Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 999–1008.

[40] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy, "Robust subspace learning: Robust PCA, robust subspace tracking, and robust subspace recovery," *IEEE Signal Process. Mag.*, vol. 35, no. 4, pp. 32–55, Jul. 2018.

[41] Y. Li, Y. Chi, H. Zhang, and Y. Liang, "Non-convex low-rank matrix recovery with arbitrary outliers via median-truncated gradient descent," *Inf. Inference, A J. IMA*, vol. 9, no. 2, pp. 289–325, Jun. 2020.

[42] I. Diakonikolas and D. M. Kane, *Algorithmic High-Dimensional Robust Statistics*. Cambridge, U.K.: Cambridge Univ. Press, 2023.

[43] I. Diakonikolas and D. M. Kane, "Recent advances in algorithmic high-dimensional robust statistics," 2019, *arXiv:1911.05911*.

[44] I. Diakonikolas, G. Kamath, D. Kane, J. Li, A. Moitra, and A. Stewart, "Robust estimators in high-dimensions without the computational intractability," *SIAM J. Comput.*, vol. 48, no. 2, pp. 742–864, Jan. 2019.

[45] K. A. Lai, A. B. Rao, and S. Vempala, "Agnostic estimation of mean and covariance," in *Proc. IEEE 57th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2016, pp. 665–674.

[46] S. Minsker, "Geometric median and robust estimation in Banach spaces," 2013, *arXiv:1308.1334*.

[47] S. Nayer, P. Narayanamurthy, and N. Vaswani, "Provable low rank phase retrieval," *IEEE Trans. Inf. Theory*, vol. 66, no. 9, pp. 5875–5903, Sep. 2020.

[48] S. Nayer and N. Vaswani, "Sample-efficient low rank phase retrieval," *IEEE Trans. Inf. Theory*, vol. 67, no. 12, pp. 8190–8206, Dec. 2021.

[49] S. Nayer and N. Vaswani, "Fast and sample-efficient federated low rank matrix recovery from column-wise linear and quadratic projections," *IEEE Trans. Inf. Theory*, vol. 69, no. 2, pp. 1177–1202, Feb. 2023.

[50] N. Vaswani, "Efficient federated low rank matrix recovery via alternating GD and minimization: A simple proof," *IEEE Trans. Inf. Theory*, vol. 70, no. 7, pp. 5162–5167, Jul. 2024.

[51] A. Grammenos, R. M. Smith, J. Crowcroft, and C. Mascolo, "Federated principal component analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6453–6464.

[52] Z. Zhang, G. Zhu, R. Wang, V. K. N. Lau, and K. Huang, "Turning channel noise into an accelerator for over-the-air principal component analysis," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 7926–7941, Oct. 2022.

[53] S. Silva, B. A. Gutman, E. Romero, P. M. Thompson, A. Altmann, and M. Lorenzi, "Federated learning in distributed medical databases: Meta-analysis of large-scale subcortical brain data," in *Proc. IEEE 16th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2019, pp. 270–274.

[54] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1989.

[55] S. X. Wu, H.-T. Wai, L. Li, and A. Scaglione, "A review of distributed algorithms for principal component analysis," *Proc. IEEE*, vol. 106, no. 8, pp. 1321–1340, Aug. 2018.

[56] R. Vershynin, *High-Dimensional Probability: An Introduction With Applications in Data Science*, vol. 47. Cambridge, U.K.: Cambridge Univ. Press, 2018.

[57] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[58] E. Weiszfeld, "Sur le point pour lequel la somme des distances de η points donnés est minimum," *Tohoku Math. J., First Ser.*, vol. 43, pp. 355–386, 1937.

[59] A. Beck and S. Sabach, "Weiszfeld's method: Old and new results," *J. Optim. Theory Appl.*, vol. 164, no. 1, pp. 1–40, 2015.

[60] M. B. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford, "Geometric median in nearly linear time," in *Proc. 48th Annu. ACM Symp. Theory Comput.*, Jun. 2016, pp. 9–21.

[61] Y. Chen, Y. Chi, J. Fan, and C. Ma, "Spectral methods for data science: A statistical perspective," *Found. Trends Mach. Learn.*, vol. 14, no. 5, pp. 566–806, 2021.

[62] C. Davis and W. M. Kahan, "The rotation of eigenvectors by a perturbation. III," *SIAM J. Numer. Anal.*, vol. 7, no. 1, pp. 1–46, Mar. 1970.

[63] M. Hardt and E. Price, "The noisy power method: A meta algorithm with applications," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2861–2869.

[64] R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices," 2010, *arXiv:1011.3027*.

[65] Y. Chen, Y. Chi, J. Fan, and C. Ma, "Spectral methods for data science: A statistical perspective," *arXiv:2012.08496*, 2020.

[66] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "DETOX: A redundancy-based framework for faster and more robust gradient aggregation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[67] M. Rudelson and R. Vershynin, "Smallest singular value of a random rectangular matrix," *Commun. Pure Appl. Math., A J. Issued Courant Inst. Math. Sci.*, vol. 62, no. 12, pp. 1707–1739, Dec. 2009.

Ankit Pratap Singh received the master's degree (Hons.) in statistics and computing from Banaras Hindu University, Varanasi, India, in 2020. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA. His research interests include federated machine learning and information theory, with a particular focus on Byzantine resilience, representation learning, and low-rank matrix recovery. He received the Gold Medal from Banaras Hindu University for his master's degree.

Namrata Vaswani (Fellow, IEEE) received the B.Tech. degree from Indian Institute of Technology (IIT-Delhi), India, in 1999, and the Ph.D. degree from the University of Maryland, College Park, in 2004.

She is currently a Professor of electrical and computer engineering and the Anderlik Professor of Engineering with Iowa State University. She is also the Director of the CyMath (Graduate Student Led) School K-8 Math Mentoring/Tutoring Program, Iowa State University. Her research interests include statistical machine learning and signal processing and their applications in medical imaging and video. She is a fellow of AAAS. She was a recipient of the IEEE Signal Processing Society (SPS) Best Paper Award in 2014, the University of Maryland ECE Distinguished Alumni Award in 2019, and Iowa State Mid-Career Achievement in Research Award in 2019. She has served as an Area Editor for *IEEE Signal Processing Magazine* and an Associate Editor for IEEE TRANSACTIONS ON INFORMATION THEORY and IEEE TRANSACTIONS ON SIGNAL PROCESSING. She has guest edited a special issue for PROCEEDINGS OF THE IEEE.