Streaming PCA and Subspace Tracking: The Missing Data Case

Laura Balzano, *Senior Member, IEEE,* Yuejie Chi, *Senior Member, IEEE,* and Yue M. Lu, *Senior Member, IEEE*

Abstract—**For many modern applications in science and engineering, data are collected in a streaming fashion carrying time-varying information, and practitioners need to process them with a limited amount of memory and computational resources in a timely manner for decision making. This often is coupled with the missing data problem, such that only a small fraction of data attributes are observed. These complications impose significant, and unconventional, constraints on the problem of streaming Principal Component Analysis (PCA) and subspace tracking, which is an essential building block for many inference tasks in signal processing and machine learning. This survey article reviews a variety of classical and recent algorithms for solving this problem with low computational and memory complexities, particularly those applicable in the big data regime with missing data. We illustrate that streaming PCA and subspace tracking algorithms can be understood through algebraic and geometric perspectives, and they need to be adjusted carefully to handle missing data. Both asymptotic and non-asymptotic convergence guarantees are reviewed. Finally, we benchmark the performance of several competitive algorithms in the presence of missing data for both well-conditioned and ill-conditioned systems.**

Index Terms—**subspace tracking, streaming PCA, subspace and low-rank models, missing data, ODE analysis**

I. INTRODUCTION

The explosion of data collection across a variety of domains, for purposes that range from scientific to commercial to policy-oriented, has created a data deluge that requires new tools for extracting useful insights from data. Principal Component Analysis (PCA) [1] and subspace tracking are arguably some of the most commonly used tools for exploring and understanding data. The fundamental mathematics and algorithms for identifying signal subspaces from data have been studied for nearly a century. However, in the modern context, many novel challenges arise, due to the severe mismatch between limited resources available at computational platforms and increasing demand of processing high-dimensional data. In particular, this survey article is motivated by the following aspects of modern data processing.

- *Large-scale and high-rate.* Data are collected at an extremely large scale with many variables, such as in video surveillance or internet monitoring, and they can accumulate at such high rates that real-time processing is

L. Balzano is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA (e-mail: girasole@umich.edu).

Y. Chi is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA (e-mail: yuejiechi@cmu.edu).

Y. M. Lu is with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA (e-mail: yuelu@seas.harvard.edu).

necessary for timely decision making. Therefore, classical batch algorithms for data processing are replaced by online, streaming algorithms that have much smaller memory and computational footprints.

- *Missing data.* At each time instance, only a very small subset of the data attributes may be measured, due to hardware limitations, power constraints, privacy concerns, or simple lack of observations. Therefore, classical algorithms that do not account for missing data may yield highly sub-optimal performance and need to be redesigned.

To elaborate on these modern challenges, we describe two concrete examples in more detail. First, consider recommendation systems [2], where users' past product use and opinions are collected. Based on such data, the system attempts to predict other products of interest to those (and potentially other) users. This is of course a scenario involving extremely sparse observations in high dimensions—a user has only purchased or rated a vanishingly small number of products from a company. Moreover, as the users rate more products and as new products become available, it is desirable to update the system's predictions on user preference in an online manner.

As another example, consider the rigid structure from motion problem in computer vision [3], [4]. One seeks to build a 3D model of a scene based on a sequence of 2D images that, for an orthographic camera, are projections of that scene onto a plane. Features in the scene can be tracked through the images, and a matrix of their locations in the images has a low-rank (3-dimensional) factorization in terms of the true 3D locations of feature points and the locations of the cameras at each image frame. The problem is obviously high dimensional, and it is also natural to consider the streaming setting, as large numbers of features can be tracked across image frames that arrive sequentially at a high rate. Moreover, many points in the scene are not visible in all image frames due to occlusion. Therefore, while the low-rank subspace of the data recovers the 3D structure of the entire scene, one must estimate this subspace in the presence of missing data.

The list of modern applications continues. The question is: *can we have scalable and accurate algorithms for subspace learning that work well even in the presence of missing data in a dynamic environment?*

A. Subspace Models and Missing Data

Subspace models have long been an excellent model for capturing intrinsic, low-dimensional structures in large datasets. A celebrated example, PCA [1], has been successfully applied to

many signal processing applications including medical imaging, communications, source localization and clutter tracking in radar and sonar, computer vision for object tracking, system identification, traffic data analysis, and speech recognition, to name just a few. The calculated principal components and best-fit subspaces to a dataset not only allow dimensionality reduction but also provide intermediate means for signal estimation, noise removal, and anomaly detection [5]. As we highlight in this paper, the principal components can be updated using incoming data in a streaming manner, thus offering tracking capabilities that are necessary for real-time decision making.

While there are a plethora of traditional algorithms for performing PCA on a batch dataset and for estimating and tracking the principal components in a streaming scenario (see, *e.g.*, [6], for a survey of earlier literature), most of these algorithms were developed during a time when datasets of interest had a moderate number of variables (say 10-100) and were collected in a controlled environment with little or no missing entries. As argued earlier, modern datasets are being collected on vast scales, in a much less controlled way, often with overwhelmingly many missing entries. In light of this prevalent and modern challenge in signal processing and machine learning, classical algorithms must be adjusted in order to gracefully handle missing data.

When do we have hope to recover missing data? If the complete high-dimensional data are well-approximated by their projection onto a lower-dimensional subspace, and hence in some sense redundant, then it is conceivable that incomplete or subsampled data may provide sufficient information for the recovery of that subspace. A related problem in the batch setting is the celebrated problem of low-rank matrix completion [7], [8], which suggests that it is possible to recover a highly incomplete matrix if its rank is much smaller than the dimension. This is the central intuition that motivates work on *streaming PCA and subspace tracking with missing data*. A burst of research activity has been devoted to developing algorithms and theoretical underpinnings for this problem over the last several years in signal processing, machine learning, and statistics. Moreover, powerful results from random matrix theory and stochastic processes have been leveraged to develop performance guarantees for both traditional and newly proposed methods. At the same time, these methods are also finding new applications to emerging data science applications such as monitoring of smart infrastructures [9], neurological, and physiological signal processing and understanding [10].

B. Overview of Subspace Tracking Algorithms

There is a long history of subspace tracking algorithms in the literature of signal processing. An extensive survey of methods prior to 1990 was provided in a popular Proceedings of the IEEE article by Comon and Golub [6]. As the common problem dimension was relatively small at that time, the focus was mostly on performance and computational complexity for fully observed data of moderate dimensions. Since then, new algorithms have been and continue to be developed with a focus on minimizing computation and memory complexity for very high-dimensional problems with missing data, without

suffering too much on performance [11]. Consider the problem of estimating or tracking a k-dimensional subspace in $\mathbb{R}^d$, where $k \ll d$. For modern applications, it is desirable that both the computational complexity (per update) and the memory complexity scale at most linearly with respect to d. Moreover, modern applications may require the algorithm to handle a range of missing data, from just a small fraction of missing entries to the information-theoretic limit of only $\mathcal{O}(k \log d)$ entries observed in each data vector[1].

Broadly speaking, there are two perspectives from which researchers have developed and studied streaming PCA and subspace tracking algorithms, as categorized by Smith [14]. The first class of algorithms can be interpreted through an algebraic lens; these can be regarded as variants of incremental methods for calculating top-k eigenvectors or singular vectors of a time-varying matrix, such as the sample covariance matrix. Since this time-varying matrix is typically updated by a rank-one modification, various matrix manipulation techniques can be exploited to reduce computational and memory complexities. This viewpoint is particularly useful for understanding algorithms such as Incremental SVD [15], Karasalo's method [16], Oja's method [17], Krasulina's method [18], [19], and other algorithms based on power iterations [20], [21], to name a few.

The other class of algorithms can be interpreted through a geometric lens. These algorithms are constructed as the solution to the optimization of a certain loss function, *e.g.*, via gradient descent, designed in either Euclidean space or on a matrix manifold such as the Grassmannian. We focus mainly on methods where the loss function is updated by one additional term per streaming column vector, and the previous estimate can be used as a warm start or initialization. This viewpoint is particularly useful in the presence of missing data, since missing data are easily incorporated into a loss function, and has therefore been leveraged more often than the algebraic viewpoint in the design of subspace tracking algorithms that are tolerant to missing data. Examples include GROUSE [22], PETRELS [23], ReProCS [24], PAST [25], online nuclear norm minimization [26], and other algorithms based on stochastic approximation [27], to name a few.

The two classes of algorithms, while having distinct features, can often be unified, as an algorithm can often be interpreted through both perspectives. The trade-off between convergence speed in static environments and tracking speed in dynamic environments is also an important consideration in practice, achieved by balancing the influence from historical data and current data. This can be done by discounting historical data in the construction of the time-varying matrix in algebraic methods, and in the construction of the loss function or selection of step sizes in geometric methods.

There is also a vast literature on establishing theoretical performance guarantees for various streaming PCA and subspace tracking algorithms. Classical analysis is primarily done in the asymptotic regime (see, *e.g.*, [28], [29]), where the discrete-

---

[1]This is the information-theoretic lower bound of measurements for an arbitrary incoherent rank-k matrix when entries from about d total column vectors are observed uniformly at random [12]. For a generic matrix, we need only $\mathcal{O}(\max(k, \log d))$ entries per column with $\mathcal{O}(kd)$ total columns [13].

time stochastic processes associated with the algorithms are shown to converge, in the scaling limit [30], [31], to the solution of some deterministic differential equations. Recent developments in performance analysis include new and more tractable asymptotic analysis for high-dimensional cases [32]–[35], as well as finite-sample probabilistic performance guarantees [36]–[40].

C. Organization of the Paper

We first describe in Section II the problem formulation of PCA and streaming PCA in the presence of missing data. We then survey algorithms that perform streaming subspace estimation and tracking with full or incompletely observed columns: Section III focuses on those using algebraic approaches and Section IV on those using geometric approaches. Many of these algorithms have associated theoretical analysis with regards to the estimation accuracy and algorithmic convergence rates, which we discuss in Section V. We then provide numerical comparisons of a number of competitive algorithms in Section VI and conclude in Section VII.

D. Notations

Throughout this paper, we use boldface letters to denote vectors and matrices, *e.g.*, $\boldsymbol{a}$ and $\boldsymbol{A}$. For a positive semidefinite (PSD) matrix $\boldsymbol{A}$, we write $\boldsymbol{A} \succeq 0$. The transpose of $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^T$, and $\|\boldsymbol{A}\|$, $\|\boldsymbol{A}\|_{\mathrm{F}}$, and $\mathrm{Tr}(\boldsymbol{A})$ denote the spectral norm, the Frobenius norm and the trace, respectively. The expectation of a random variable a is written as $\mathbb{E}[a]$. The identity matrix of dimension k is written as $\boldsymbol{I}_k$. We shall use d to denote the dimension of the fully observed data vector and k to denote the dimension of the subspace to be estimated. A subscript n on the data vector $\boldsymbol{x}_n \in \mathbb{R}^d$ refers to its order in a sequence of vectors, and the notation $\boldsymbol{x}_n(i)$ refers to the ith component of the vector $\boldsymbol{x}_n$.

II. PROBLEM FORMULATION

In this section, we will start by formulating the problem of subspace estimation in the batch setting, which serves as a good starting point to motivate streaming PCA and subspace tracking in the streaming setting with missing data.

A. PCA in the Batch Setting

The PCA or subspace estimation problem can be formulated either *probabilistically*, where data are assumed to be random vectors drawn from a distribution with mean zero and some covariance matrix whose principal subspace we wish to estimate, or *deterministically*, where we seek the best rank-k subspace that fits the given data. Both models are used extensively throughout the literature. The former is used more prevalently in the signal processing and statistics literature, while the latter is more prevalent in applied mathematics, optimization, and computer science literature. The problem formulations result in equivalent optimization problems, and so we put them here together for a unified view.

(a) *Probabilistic view*: Consider a stationary, d-dimensional random process $\boldsymbol{x} \in \mathbb{R}^d$, which has a zero mean and a covariance matrix $\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T]$.[2] Denote the eigenvalue decomposition (EVD) of the covariance matrix as $\boldsymbol{\Sigma} = \widetilde{\boldsymbol{U}}\boldsymbol{\Lambda}\widetilde{\boldsymbol{U}}^T$, where $\widetilde{\boldsymbol{U}} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d]$ has orthonormal columns, and $\boldsymbol{\Lambda} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_d\}$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$ are the eigenvalues arranged in a non-increasing order. Our goal is to estimate the top-k eigenvectors, also called the principal components $\boldsymbol{U}^* = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$, of $\boldsymbol{\Sigma}$, given a finite number of i.i.d. data samples, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \sim \boldsymbol{x}$. Note that we do not require $\boldsymbol{\Sigma}$ be a rank-k matrix.

(b) *Deterministic view*: In a deterministic formulation, the data samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ are considered arbitrary. We wish to find the rank-k subspace that best fits these data in the sense of minimizing the ℓ_2 projection error, that is

$$\widehat{\boldsymbol{U}}_n = \operatorname*{argmin}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}, \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_k} \sum_{\ell=1}^n \left\| \boldsymbol{x}_\ell - \mathcal{P}_{\boldsymbol{U}}(\boldsymbol{x}_\ell) \right\|_2^2 \quad (1)$$

$$= \operatorname*{argmin}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}, \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_k} \left\| \boldsymbol{X}_n - \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{X}_n \right\|_{\mathrm{F}}^2 \quad (2)$$

$$= \operatorname*{argmax}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}, \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_k} \mathrm{Tr}\left(\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{\Sigma}_n \right), \quad (3)$$

where $\mathcal{P}_{\boldsymbol{U}}$ denotes the projection operator onto the column span of the matrix $\boldsymbol{U}$ and $\mathcal{P}_{\boldsymbol{U}} = \boldsymbol{U}\boldsymbol{U}^T$ when $\boldsymbol{U}$ has orthonormal columns, $\boldsymbol{X}_n = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ concatenates the data vectors as columns into a matrix, and $\boldsymbol{\Sigma}_n = \sum_{\ell=1}^n \boldsymbol{x}_\ell \boldsymbol{x}_\ell^T = \boldsymbol{X}_n\boldsymbol{X}_n^T$ is the (unscaled) Sample Covariance Matrix (SCM).

The equivalence of (2) and (3) suggests that finding the subspace that maximizes the explained variance of $\boldsymbol{\Sigma}_n$ is equivalent to minimizing the approximation error of the data matrix $\boldsymbol{X}_n$. While the formulations (2) or (3) are non-convex, both due to the cost function's non-convexity in $\boldsymbol{U}$ and the non-convex constraint $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_k$, they admit a well-defined solution, solved by the SVD of $\boldsymbol{X}_n$, equivalently the EVD of $\boldsymbol{\Sigma}_n$, as was discovered independently by [42] (see [43], [44] for details) and [45]. Specifically, the solution $\widehat{\boldsymbol{U}}_n$ is given as the top-k eigenvectors of the SCM $\boldsymbol{\Sigma}_n$.

(c) *Unified perspective*: Consider the following expected loss function

$$J(\boldsymbol{U}) = \mathbb{E} \left\| \boldsymbol{x} - \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{x} \right\|_2^2, \quad (4)$$

where $\boldsymbol{U} \in \mathbb{R}^{d \times k}$ and the expectation is taken with respect to $\boldsymbol{x}$ which is zero mean with a covariance matrix $\boldsymbol{\Sigma}$. The following important result was proven in [25]: if $\lambda_k > \lambda_{k+1}$, *i.e.*, if there is a strict eigengap, then the global optima of $J(\boldsymbol{U})$ correspond to $\boldsymbol{U}$ that contains the top-k eigenvectors of $\boldsymbol{\Sigma}$ up to an orthonormal transformation, matching the solution of PCA in the probabilistic view. Interestingly, the solution to the deterministic formulation (1) can be thought of as an empirical version of (4), if the data samples are indeed drawn according to the probabilistic model. Moreover, in this case, $\widehat{\boldsymbol{U}}_n$ produces an order-wise near-optimal estimate to $\boldsymbol{U}^*$ for a large family of distributions [46]. In this regard, the two formulations are equivalent in some sense, though in the deterministic setting, there need not be any generative model or "ground truth" for the underlying subspace.

---

[2]It is straightforward to consider the complex-valued case $\boldsymbol{x} \in \mathbb{C}^d$, but we only consider the real case in this survey for simplicity. For more information, see [41].

B. Streaming PCA and Subspace Tracking

In a streaming setting, the data samples arrive sequentially over time, with each sample only seen once,[3] and one wishes to update the subspace estimate sequentially without accessing historical data. In a dynamic environment, either the covariance matrix or the best rank-k subspace can be time-varying — therefore, we wish to track such changes as quickly as possible.

In this survey article, we use the terminology "streaming PCA" and "subspace tracking" interchangeably to refer to algorithms that can update and track a data subspace using streaming observations. Nonetheless, we acknowledge they have different connotations and indeed they have arisen from different contexts. The terminology "subspace tracking" is common in the literature of signal processing [6], where one often needs to update the subspace in a dynamic environment as in array signal processing or communications. The more recent terminology of "online PCA" or "streaming PCA" can be found in the machine learning literature, motivated by the study in computer science of trying to replicate the behavior of batch PCA with streaming data [47] or data too large for memory. In addition, "incremental SVD" [15] or "updating the SVD" [48] are terminology used more classically in numerical methods. It turns out that all of the algorithms reviewed herein can handle both the settings where the underlying subspace is static or time-varying by adjusting parameters within the algorithm such as data discounting factors or step sizes.

Streaming PCA can be considered as a nonconvex stochastic approximation problem, given by (4). The solution to the batch problem that we outlined in Section II-A is no longer appropriate for the streaming setting — it requires one to formulate and store the SCM $\boldsymbol{\Sigma}_n$, which has a memory complexity of $\mathcal{O}(d^2)$, and to estimate the top-k eigenvectors directly from the SCM, which has a time complexity of $\mathcal{O}(nd^2)$. Both these memory and time complexities are too expensive for large-scale problems. It is greatly desirable to have algorithms with computation and memory complexities that grow at most linearly in d.

C. Missing Data

An important setting that we will consider in this survey is *missing data*, where only a subset of the coordinates of of each sample $\boldsymbol{x}_n$ are observed. We denote this measurement as

$$\mathcal{P}_{\Omega_n}(\boldsymbol{x}_n), \qquad (5)$$

where $\mathcal{P}_{\Omega_n}$ is a projection operator onto the coordinates represented by an observation mask, $\Omega_n \in \{0,1\}^d$, where $\boldsymbol{x}_n(i)$ (the ith entry of $\boldsymbol{x}_n$) is observed if and only if $\Omega_n(i) = 1$. This issue poses severe challenges for most PCA algorithms, particularly when the number of observed entries is much smaller than d. To begin, one may be concerned with *identifiability*: can we find a unique subspace of rank-k that is consistent with the partial observations? Luckily, the answer to this question is yes, at least in the batch setting

where the problem is equivalent to that of low-rank matrix completion: under mild assumptions, the low-rank subspace can be reconstructed from subsampled column vectors as long as there are enough observations.[4] It may also be tempting to execute subspace tracking algorithms by ignoring the missing data and padding with zeros at the missing entries, however the sample covariance matrix constructed in this way leads to a biased estimator [49], [50]. Therefore, one must think more carefully about how to handle missing data in this context.

III. ALGEBRAIC METHODS

In this section and the next, we will discuss two classes of algorithms based on algebraic approaches and geometric approaches respectively, as outlined in Section I-B. The algebraic approaches are based on finding the top eigenvectors of a recursively updated SCM, or a surrogate of it, given as

$$\boldsymbol{\Sigma}_n = \alpha_n \boldsymbol{\Sigma}_{n-1} + \beta_n \boldsymbol{x}_n \boldsymbol{x}_n^T, \qquad (6)$$

where α_n and β_n balance the contributions from the previous SCM and the current data sample. Two popular choices are equal weights on all time slots, which is

$$\alpha_n = 1, \quad \beta_n = 1;$$

and discounting on historical data, which is

$$\alpha_n = \lambda, \quad \beta_n = 1, \quad 0 \ll \lambda < 1.$$

Equivalently, the above can be reworded as finding the top singular vectors of a recursively updated data matrix $\boldsymbol{X}_n$. As we are interested in calculating or approximating the top-k eigenvectors of $\boldsymbol{\Sigma}_n$, algebraic methods use matrix manipulations and exploit the simplicity of the rank-one update to reduce complexity.

A. Incremental Singular Value Decomposition (ISVD)

We begin by discussing the ISVD approach of Bunch and Neilsen [48], which is an exact method to compute the full SVD of a streaming full data matrix, *i.e.*, with sequentially arriving, full data vectors. This algorithm is given in Algorithm 1 and is the result of some simple observations about the relationship of the SVD of $\boldsymbol{X}_n$ and that of $\boldsymbol{X}_{n-1}$. Suppose we are given the compact SVD of the data matrix at time $n-1$,

$$\boldsymbol{X}_{n-1} = \begin{bmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_{n-1} \end{bmatrix} = \widetilde{\boldsymbol{U}}_{n-1} \widetilde{\boldsymbol{S}}_{n-1} \widetilde{\boldsymbol{V}}_{n-1}^T,$$

where $\widetilde{\boldsymbol{U}}_{n-1} \in \mathbb{R}^{d \times d}$ and $\widetilde{\boldsymbol{V}}_{n-1} \in \mathbb{R}^{(n-1) \times (n-1)}$ are orthonormal, and $\widetilde{\boldsymbol{S}}_{n-1} \in \mathbb{R}^{d \times (n-1)}$ is the concatenation of two matrices: a diagonal matrix (of size $\min\{d, n-1\}$) with non-negative non-increasing diagonal entries, and an all-zero matrix. Note that we are using the $\widetilde{\boldsymbol{U}}$ notation for the square orthogonal matrix as opposed to $\boldsymbol{U}$ for a $d \times k$ matrix, as in Section II-A, because we are computing the full SVD, not a low-rank approximation. For simplicity of exposition, let's assume $d \geq n$ (but both cases $d < n$ and $d \geq n$ are described in Algorithm 1). We wish to compute the SVD of

$$\boldsymbol{X}_n = \begin{bmatrix} \boldsymbol{X}_{n-1} & \boldsymbol{x}_n \end{bmatrix} = \widetilde{\boldsymbol{U}}_n \widetilde{\boldsymbol{S}}_n \widetilde{\boldsymbol{V}}_n^T,$$

---

[3]This is different from what is known as a stochastic setting, where samples may be accessed at multiple times or in multiple passes.

[4]"Enough" observations per vector here will depend both on the number of vectors and the conditioning or coherence [7] of the subspace.

where $\widetilde{U}_n$, $\widetilde{S}_n$, and $\widetilde{V}_n$ are defined similarly as $\widetilde{U}_{n-1}$, $\widetilde{S}_{n-1}$, and $\widetilde{V}_{n-1}$.

Recognizing that

$$X_n X_n^T = X_{n-1} X_{n-1}^T + x_n x_n^T$$

and

$$\widetilde{U}_{n-1}^T X_n X_n^T \widetilde{U}_{n-1} = \widetilde{S}_{n-1} \widetilde{S}_{n-1}^T + z_n z_n^T$$

where $z_n = \widetilde{U}_{n-1}^T x_n$, we can compute the new singular values by finding the eigenvalues of $\widetilde{S}_{n-1} \widetilde{S}_{n-1}^T + z_n z_n^T$ using the zeros of the characteristic equation [51], which in this case has a special structure; in particular, if $\widetilde{\sigma}_i$ are the diagonal values of $\widetilde{S}_{n-1}$, then the zeros of

$$1 + \sum_{i=1}^d \frac{z_n(i)^2}{\widetilde{\sigma}_i^2 - \lambda} \qquad (7)$$

with respect to the variable λ identify the eigenvalues of $\widetilde{S}_{n-1} \widetilde{S}_{n-1}^T + z_n z_n^T$. Denote the resulting eigenvalues as λ_i for $i = 1, \dots, d$. To update the left singular vectors to the new $\widetilde{U}_n = \begin{bmatrix} u_1 & \dots & u_d \end{bmatrix}$, we need to solve $\left(\widetilde{S}_{n-1} \widetilde{S}_{n-1}^T + z_n z_n^T \right) u_i = \lambda_i u_i$ and normalize the solution. Therefore [6], [51],

$$u_i = \frac{(\widetilde{S}_{n-1} \widetilde{S}_{n-1}^T - \lambda_i I)^{-1} z_n}{\left\| (\widetilde{S}_{n-1} \widetilde{S}_{n-1}^T - \lambda_i I)^{-1} z_n \right\|}, \quad i = 1, \dots, d. \qquad (8)$$

So far, the above derivations assume $\widetilde{U}_{n-1}$ is a square orthonormal matrix, and the resulting computations are suitable for incremental updates of the full SVD. However, this still requires $\mathcal{O}(dn^2 + n^3)$ complexity[5] for *every iteration* computing the full SVD (*i.e.*, all singular values and left/right singular vectors) using Algorithm 1, and the memory requirement grows as n grows if the data are full rank (or low rank with even a very small amount of additive noise), which are both undesirable.

On the other hand, estimating a thin SVD or the top k-dimensional singular subspace can improve computation. In fact, if X_{n-1} is exactly rank-k, this incremental approach requires fewer computations as pointed out in [52]. In this case, take the first k columns of $\widetilde{U}_{n-1}$ and call these U_{n-1}. We only need these to represent X_{n-1} because the others correspond to zero singular values. Let S_{n-1} and V_{n-1} be the corresponding matrices for this thin SVD so that $X_{n-1} = U_{n-1} S_{n-1} V_{n-1}^T$. We then notice as in [52], [53] that

$$X_n = \begin{bmatrix} U_{n-1} & \frac{r_n}{\|r_n\|} \end{bmatrix} \begin{bmatrix} S_{n-1} & w_n \\ 0 & \|r_n\| \end{bmatrix} \begin{bmatrix} V_{n-1}^T & 0 \\ 0 & 1 \end{bmatrix} \qquad (12)$$

where $w_n = U_{n-1}^T x_n$ are the projection weights onto the span of this now tall matrix U_{n-1} and $r_n = x_n - U_{n-1} U_{n-1}^T x_n$ is the residual from the projection. We only must diagonalize the center matrix of (12) to find the SVD of X_n. We only assumed X_{n-1} is rank-k, but then to make this assumption at every step

---

^{5}This complexity is assuming $d > n$. In this case, the bottleneck is updating $\widetilde{U}_n$ in (10), which needs $\mathcal{O}(dn^2)$ operations for a naïve matrix multiplication; then to update $\widetilde{V}_n$, we require $\mathcal{O}(n^3)$ operations for the same step.

Algorithm 1 ISVD

1: Given x_1, set $\widetilde{U}_1 = x_1 / \|x_1\|$, $\widetilde{S}_1 = \|x_1\|$, $\widetilde{V}_1 = 1$;
2: Set $n = 2$;
3: **repeat**
4:    Define $w_n := \widetilde{U}_{n-1}^T x_n$;
5:    Define $p_n := \widetilde{U}_{n-1} w_n$; $r_n := x_n - p_n$;
6:    **if** $\|r_n\| \neq 0$ **then**
7:       Compute the SVD of the update matrix:

$$\begin{bmatrix} \widetilde{S}_{n-1} & w_n \\ 0 & \|r_n\| \end{bmatrix} = \widehat{U} \widehat{S} \widehat{V}^T, \qquad (9)$$

      by solving (7) for λ, which gives the diagonal entries of $\widehat{S}$, where $\widetilde{\sigma}_i^2$ are the diagonal entries of $\widetilde{S}_{n-1}$ and $z_n = \begin{bmatrix} w_n^T & \|r_n\| & 0 \end{bmatrix}^T$, and then solving (8) for $\widehat{U}$.

8:    Set

$$\widetilde{U}_n := \begin{bmatrix} \widetilde{U}_{n-1} & \frac{r_n}{\|r_n\|} \end{bmatrix} \widehat{U}, \quad \widetilde{S}_n := \widehat{S}. \qquad (10)$$

$$\widetilde{V}_n := \begin{bmatrix} \widetilde{V}_{n-1} & 0 \\ 0 & 1 \end{bmatrix} \widehat{V}.$$

9:    **else if** $\|r_n\| = 0$ **then**
10:      (this happens when $n > d$ or $x_n \in \text{span}\{U_{n-1}\}$)
11:     Compute the SVD of the update matrix:

$$\begin{bmatrix} \widetilde{S}_{n-1} & w_n \end{bmatrix} = \widehat{U} \widehat{S} \widehat{V}^T, \qquad (11)$$

      by solving (7) for λ, which gives the diagonal entries of $\widehat{S}$, where $\widetilde{\sigma}_i^2$ are the diagonal entries of $\widetilde{S}_{n-1}$ and $z_n = \begin{bmatrix} w_n^T & 0 \end{bmatrix}^T$, and then solving (8) for $\widehat{U}$.

12:    Set

$$\widetilde{U}_n := \widetilde{U}_{n-1} \widehat{U}, \quad \widetilde{S}_n := \widehat{S}.$$

$$\widetilde{V}_n := \begin{bmatrix} \widetilde{V}_{n-1} & 0 \\ 0 & 1 \end{bmatrix} \widehat{V}.$$

13:    **end if**
14:    $n := n + 1$;
15: **until** termination

---

is a strong assumption and means the matrix is exactly low-rank. However, this technique is used successfully as a heuristic, by truncating smaller singular values and corresponding singular vectors, even when this assumption does not hold. This method is described in the following section. Finally, we point out that Karasalo's subspace averaging algorithm [16] is similar to ISVD, but it uses specific information about the noise covariance.

B. MD-ISVD, Brand's Algorithm and PIMC

A major drawback of many linear algebraic techniques is their inapplicability to datasets with missing data. While it is not straightforward how to adapt the ISVD for missing data, there are several different approaches in the literature [3], [15], [54]. All approaches begin in the same way, by considering

Algorithm 2 MD-ISVD, Brand's algorithm, PIMC

1: Given an orthonormal matrix $\boldsymbol{U}_0 \in \mathbb{R}^{d \times k}$, $\boldsymbol{S}_0 = 0$;
2: For PIMC, $\gamma_0 = 1$.
3: Set $n = 1$;
4: **repeat**
5:     Define $\boldsymbol{w}_n := \arg\min_{\boldsymbol{w}} \|\mathcal{P}_{\Omega_n} (\boldsymbol{U}_{n-1}\boldsymbol{w} - \boldsymbol{x}_n)\|_2^2$;
6:     Define $\boldsymbol{p}_n := \boldsymbol{U}_{n-1}\boldsymbol{w}_n$;

$$\boldsymbol{r}_n(i) = \begin{cases} \boldsymbol{x}_n(i) - \boldsymbol{p}_n(i) & \Omega_n(i) = 1 \\ 0 & \text{otherwise} \end{cases}.$$

7:     Compute the SVD of the update matrix:

$$\begin{bmatrix} \boldsymbol{\Gamma}_{n-1} & \boldsymbol{w}_n \\ \boldsymbol{0} & \|\boldsymbol{r}_n\| \end{bmatrix} = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{S}}\widehat{\boldsymbol{V}}^T, \quad (13)$$

    where

$$\boldsymbol{\Gamma}_{n-1} = \begin{cases} \boldsymbol{S}_{n-1} & \text{for MD-ISVD,} \\ \lambda\boldsymbol{S}_{n-1} & \text{for Brand's algorithm,} \\ \frac{\gamma_n}{\|\boldsymbol{S}_{n-1}\|_{\mathrm{F}}}\boldsymbol{S}_{n-1}, & \text{for PIMC,} \end{cases}$$

    with $\gamma_n^2 = \gamma_{n-1}^2 + \|\mathcal{P}_{\Omega_n}(\boldsymbol{x}_n)\|_2^2$ for PIMC.
8:     Set $\check{\boldsymbol{U}}_n := \begin{bmatrix} \boldsymbol{U}_{n-1} & \frac{\boldsymbol{r}_n}{\|\boldsymbol{r}_n\|} \end{bmatrix} \widehat{\boldsymbol{U}}$;
9:     Set $\boldsymbol{U}_n$ as the first k columns of $\check{\boldsymbol{U}}_n$ and $\boldsymbol{S}_n$ as the top k-by-k block of $\widehat{\boldsymbol{S}}$.
10:     $n := n + 1$;
11: **until** termination

---

how to compute two key quantities, the projection weights $\boldsymbol{w}_n$ and the residual $\boldsymbol{r}_n$, given missing data. Whereas for complete data, we have $\boldsymbol{w}_n = \operatorname{argmin}_{\boldsymbol{w}} \|\boldsymbol{x}_n - \boldsymbol{U}_{n-1}\boldsymbol{w}\|_2^2 = \boldsymbol{U}_{n-1}^T\boldsymbol{x}_n$, for missing data one may solve

$$\boldsymbol{w}_n = \operatorname*{argmin}_{\boldsymbol{w}} \|\mathcal{P}_{\Omega_n} (\boldsymbol{x}_n - \boldsymbol{U}_{n-1}\boldsymbol{w})\|_2^2 \quad (14)$$

where $\mathcal{P}_{\Omega_n}$ is the measurement operator with missing data as in (5). Then letting $\boldsymbol{p}_n = \boldsymbol{U}_{n-1}\boldsymbol{w}_n$, define the residual to be

$$\boldsymbol{r}_n(i) = \begin{cases} \boldsymbol{x}_n(i) - \boldsymbol{p}_n(i) & \text{if } \Omega_n(i) = 1 \\ 0 & \text{otherwise} \end{cases}. \quad (15)$$

All the methods we describe in this section use these quantities in place of $\boldsymbol{w}_n$ and $\boldsymbol{r}_n$ in Algorithm 1. They also all mimic (9) for the update, and once they have observed enough vectors they *truncate* the $k + 1$ singular value and corresponding singular vectors.

The methods diverge only in the way they replace the singular value matrix $\boldsymbol{S}_{n-1}$ in (9). Brand [15] replaces $\boldsymbol{S}_{n-1}$ with $\lambda\boldsymbol{S}_{n-1}$ where $0 \ll \lambda < 1$ is a scalar weight that diminishes the influence of previous singular values. If one takes $\lambda = 1$, this is arguably the most direct extension of ISVD to missing data, and following [3] we call this algorithm Missing Data-ISVD (MD-ISVD). Kennedy et al. [54] present a Polar Incremental Matrix Completion (PIMC) method, which weights $\boldsymbol{S}_{n-1}$ with a scalar based on the norm of the data observed thus far. These are different approaches to modeling the uncertainty in the singular values arising from incomplete observations. These algorithms are together summarized in

---

Algorithm 3 Oja's algorithm with missing data

1: Given an orthonormal matrix $\boldsymbol{U}_0 \in \mathbb{R}^{d \times k}$;
2: Set $n := 1$;
3: **repeat**
4:     Define $\boldsymbol{w}_n := \operatorname{argmin}_{\boldsymbol{w}} \|\mathcal{P}_{\Omega_n} (\boldsymbol{x}_n - \boldsymbol{U}_{n-1}\boldsymbol{w})\|_2^2$;
5:     Set $\boldsymbol{p}_n = \boldsymbol{U}_{n-1}\boldsymbol{w}_n$.
6:     Set $\tilde{\boldsymbol{x}}_n = \begin{cases} \boldsymbol{x}_n(i) & \text{if } \Omega_n(i) = 1 \\ \boldsymbol{p}_n(i) & \text{otherwise} \end{cases}$
7:     $\boldsymbol{U}_n = \Pi(\boldsymbol{U}_{n-1} + \eta_n\tilde{\boldsymbol{x}}_n\boldsymbol{w}_n^T)$,
8:     $n := n + 1$;
9: **until** termination

Algorithm 2. These different rules provide different trade-offs in (6), as they represent different weighting schemes on historical data.

C. Oja's method

Oja's method was originally proposed in 1982 [17]. It is a very popular method for streaming PCA, and recent attention has yielded significant insight into its practical performance (see discussion in Section V). Given an orthonormal initialization $\boldsymbol{U}_0 \in \mathbb{R}^{d \times k}$, at the nth time Oja's method updates to $\boldsymbol{U}_n$ according to the input data $\boldsymbol{x}_n$ as

$$\boldsymbol{U}_n = \Pi(\boldsymbol{U}_{n-1} + \eta_n\boldsymbol{x}_n\boldsymbol{x}_n^T\boldsymbol{U}_{n-1}), \quad (16)$$

where $\Pi(\boldsymbol{W}) = \boldsymbol{Q}$ is an orthogonalization operator, *i.e.*, $\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{R}$ is the QR decomposition. The parameter η_n is the step size or learning rate that may change with time.

While Oja's method has not been derived for the missing data case in the literature, following our discussion on ISVD, one realizes that if as before we let $\boldsymbol{w}_n = \boldsymbol{U}_{n-1}^T\boldsymbol{x}_n$ be the coefficient of $\boldsymbol{x}_n$ in the previous estimate $\boldsymbol{U}_{n-1}$, then Oja's method is equivalent to

$$\boldsymbol{U}_n = \Pi(\boldsymbol{U}_{n-1} + \eta_n\boldsymbol{x}_n\boldsymbol{w}_n^T). \quad (17)$$

A straightforward extension in the missing data case is then to estimate the coefficient $\boldsymbol{w}_n$ as (14), and to fill in the missing entries in $\boldsymbol{x}_n$ as follows. Let $\boldsymbol{p}_n = \boldsymbol{U}_{n-1}\boldsymbol{w}_n$, and the data vector can be interpolated as

$$\tilde{\boldsymbol{x}}_n = \begin{cases} \boldsymbol{x}_n(i) & \text{if } \Omega_n(i) = 1 \\ \boldsymbol{p}_n(i) & \text{otherwise} \end{cases}.$$

Then Oja's update rule in the missing data case becomes

$$\boldsymbol{U}_n = \Pi(\boldsymbol{U}_{n-1} + \eta_n\tilde{\boldsymbol{x}}_n\boldsymbol{w}_n^T). \quad (18)$$

This algorithm is summarized in Algorithm 3. Note that Oja's original method with full data becomes a special case of this update. We study this extension in the numerical experiments reported in Section VI.

Finally, we note that closely related to Oja's is another method called Krasulina's algorithm [19], which is developed for updating a rank-1 subspace with full data:

$$\boldsymbol{U}_n = \boldsymbol{U}_{n-1} + \eta_n \left(\boldsymbol{x}_n\boldsymbol{x}_n^T - \frac{\boldsymbol{U}_{n-1}^T\boldsymbol{x}_n\boldsymbol{x}_n^T\boldsymbol{U}_{n-1}}{\|\boldsymbol{U}_{n-1}\|^2}\boldsymbol{I}_d \right) \boldsymbol{U}_{n-1}. \quad (19)$$

It can be viewed as a stochastic gradient descent method with the Rayleigh quotient as its objective. Oja's method is equivalent to Krasulina's method up to the second order terms [18], [55].

Remark 1 (Block Power Method). *A block variant of Oja's method has been developed in the literature [36], [37], [56], where it partitions the input into blocks and each time processes one block in a way similar to Oja's method. These methods are referred to as the block power method, or block Oja's method, or the noisy power method. They are easier to analyze but yield suboptimal performance [40].*

IV. GEOMETRIC METHODS

In this section, we review subspace tracking algorithms developed via geometric approaches. These are developed by optimizing certain loss functions over $d \times k$ matrices in Euclidean space or the Grassmann manifold of rank-k subspaces in $\mathbb{R}^d$. Subspace tracking is enabled by optimizing a recursively updated loss function, such as the squared projection loss onto the subspace, as

$$F_n(\boldsymbol{U}) = \alpha_n F_{n-1}(\boldsymbol{U}) + \beta_n \left\| \boldsymbol{x}_n - \mathcal{P}_{\boldsymbol{U}}(\boldsymbol{x}_n) \right\|_2^2, \qquad (20)$$

where $\boldsymbol{U} \in \mathbb{R}^{d \times k}$, and n is the time index, which is typically updated by using the previous estimate as a warm start. Similarly, the choice of α_n and β_n balances the convergence rate (how fast it converges with data from a static subspace) and the tracking capability (how fast it can adapt to changes in the subspace). Additionally, the step size of some gradient algorithms can also be used as a tuning knob for tracking; a more aggressive step size will adapt more quickly to new data. Given the necessity of scalable and memory-efficient algorithms, first-order and second-order stochastic gradient descent [57] are gaining a lot of popularity recently in signal processing and machine learning.

A. GROUSE

Grassmannian Rank-One Update Subspace Estimation (GROUSE) was first introduced in [22] as an incremental gradient algorithm to build high quality subspace estimates from very sparsely sampled vectors, and has since been analyzed with fully sampled data [58], [59], noisy data [59], and missing or compressed data [58], [60] (see Section V). The objective function for the algorithm is given by

$$F_n(\boldsymbol{U}) = \sum_{\ell=1}^{n} \|\mathcal{P}_{\Omega_\ell}(\boldsymbol{x}_\ell - \boldsymbol{U}\boldsymbol{U}^T \boldsymbol{x}_\ell)\|_2^2, \qquad (21)$$

which is a special case of (20) with $\alpha_n = \beta_n = 1$. GROUSE implements a first-order incremental gradient procedure [61] to minimize this objective with respect to the subspace variable $\boldsymbol{U}$ constrained to the Grassmannian [62], the manifold of all subspaces with a fixed rank, given as

$$\min_{\boldsymbol{U} \in \mathbb{R}^{n \times k} : \boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{I}_k} F_n(\boldsymbol{U}).$$

GROUSE has iteration complexity $\mathcal{O}(dk + |\Omega_n|k^2)$ at the nth update and so is scalable to very high-dimensional applications. The algorithm steps are described in Algorithm 4.

The GROUSE update in (23) can also be written as:

$$\boldsymbol{U}_n = \boldsymbol{U}_{n-1} - \frac{\boldsymbol{U}_{n-1}\boldsymbol{w}_n \boldsymbol{w}_n^T}{\|\boldsymbol{w}_n\|^2} + \frac{\boldsymbol{y}_n \boldsymbol{w}_n^T}{\|\boldsymbol{y}_n\|\|\boldsymbol{w}_n\|}$$

where

$$\boldsymbol{y}_n = \cos(\theta_n)\frac{\boldsymbol{p}_n}{\|\boldsymbol{p}_n\|} + \sin(\theta_n)\frac{\boldsymbol{r}_n}{\|\boldsymbol{r}_n\|} .$$

This form makes it clear that GROUSE is simply replacing a direction in the current subspace estimate, $\boldsymbol{p}_n = \boldsymbol{U}_{n-1}\boldsymbol{w}_n$, with a new vector $\boldsymbol{y}_n$ that is a linear combination of $\boldsymbol{p}_n$ and the residual vector $\boldsymbol{r}_n$. This of course makes $\boldsymbol{y}_n$ orthogonal to the rest of $\boldsymbol{U}_{n-1}$, which is why $\boldsymbol{U}_n$ will necessarily also have orthogonal columns.

We note that, if the step size is not given, one can use the step size prescribed in (22). This step size maximizes the per-iteration improvement of the algorithm in a greedy way, but can therefore be susceptible to noise. For example, with fully observed data, this greedy step size will replace the direction $\boldsymbol{p}_n$ in the current iterate $\boldsymbol{U}_{n-1}$ with the observed data $\boldsymbol{x}_n$. If bounds on the noise variance are known, one can use the noise-dependent step-size given in [59], which decreases the step as the noise floor is reached.

Algorithm 4 GROUSE [22]

1: Given $\boldsymbol{U}_0$, an $d \times k$ orthonormal matrix, $0 < k < d$;
2: Optional input: Step size scheme $\eta_n > 0$;
3: Set $n := 1$;
4: **repeat**
5:     Define $\boldsymbol{w}_n := \operatorname{argmin}_{\boldsymbol{w}} \|\mathcal{P}_{\Omega_n}(\boldsymbol{x}_n - \boldsymbol{U}_{n-1}\boldsymbol{w})\|_2^2$;
6:     Define $\boldsymbol{p}_n := \boldsymbol{U}_{n-1}\boldsymbol{w}_n$;

$$\boldsymbol{r}_n(i) := \begin{cases} \boldsymbol{x}_n(i) - \boldsymbol{p}_n(i) & \text{if } \Omega_n(i) = 1 \\ 0 & \text{otherwise} \end{cases} .$$

7:     **if** η_n given **then**
8:         Set $\theta_n = \eta_n \|\boldsymbol{r}_n\|\|\boldsymbol{p}_n\|$.
9:     **else**
10:       Set

$$\theta_n = \arctan\left(\frac{\|\boldsymbol{r}_n\|}{\|\boldsymbol{p}_n\|}\right). \qquad (22)$$

11:     **end if**

$$\boldsymbol{U}_n = \boldsymbol{U}_{n-1} + \left(\cos(\theta_n) - 1\right)\frac{\boldsymbol{p}_n}{\|\boldsymbol{p}_n\|}\frac{\boldsymbol{w}_n^T}{\|\boldsymbol{w}_n\|}$$
$$+ \sin(\theta_n)\frac{\boldsymbol{r}_n}{\|\boldsymbol{r}_n\|}\frac{\boldsymbol{w}_n^T}{\|\boldsymbol{w}_n\|} . \qquad (23)$$

12:     $n := n + 1$;
13: **until** termination

In a follow-up work, Balzano *et al.* describe SAGE GROUSE [3], [53], which was derived in the context of Algorithm 2. SAGE GROUSE replaces $\boldsymbol{S}_{n-1}$ with an identity matrix the same size as $\boldsymbol{S}_{n-1}$, which makes the algorithm completely agnostic to singular values or the relative weight of singular vectors that have been learned. This can be considered as yet another way of modeling uncertainty in the singular values learned thus far in a streaming context. SAGE GROUSE

Algorithm 5 PAST [25]

1: Given $\boldsymbol{U}_0 \in \mathbb{R}^{d \times k}$, $\boldsymbol{R}_0 = \delta \boldsymbol{I}_k$;
2: Set $n := 1$;
3: **repeat**
4:   Define $\boldsymbol{w}_n := \boldsymbol{U}_{n-1}^T \boldsymbol{x}_n$;
5:   $\beta_n = 1 + \lambda^{-1} \boldsymbol{w}_n^T \boldsymbol{R}_{n-1} \boldsymbol{w}_n$,
6:   $\boldsymbol{v}_n = \lambda^{-1} \boldsymbol{R}_{n-1} \boldsymbol{w}_n$,
7:   $\boldsymbol{R}_n = \lambda^{-1} \boldsymbol{R}_{n-1} - (\beta_n)^{-1} \boldsymbol{v}_n \boldsymbol{v}_n^T$,
8:   $\boldsymbol{U}_n = \boldsymbol{U}_{n-1} + (\boldsymbol{x}_n - \boldsymbol{U}_{n-1} \boldsymbol{w}_n^T) \boldsymbol{R}_n \boldsymbol{w}_n$.
9:   $n := n + 1$;
10: **until** termination

Algorithm 6 PETRELS [23]

1: Given $\boldsymbol{U}_0 = [\boldsymbol{u}_0^1, \boldsymbol{u}_0^2, \cdots, \boldsymbol{u}_0^d]^T$, and $\boldsymbol{R}_0^i = \delta \boldsymbol{I}_k$, $\delta > 0$ for all $i = 1, \cdots, d$.
2: Set $n := 1$;
3: **repeat**
4:   Define $\boldsymbol{w}_n := \operatorname{argmin}_{\boldsymbol{w}} \|\mathcal{P}_{\Omega_n} (\boldsymbol{x}_n - \boldsymbol{U}_{n-1} \boldsymbol{w})\|_2^2$;
5:   **for** $i = 1, \cdots, d$ **do**
6:     $\beta_n^i = 1 + \lambda^{-1} \boldsymbol{w}_n^T \boldsymbol{R}_{n-1}^i \boldsymbol{w}_n$,
7:     $\boldsymbol{v}_n^i = \lambda^{-1} \boldsymbol{R}_{n-1}^i \boldsymbol{w}_n$,
8:     $\boldsymbol{R}_n^i = \lambda^{-1} \boldsymbol{R}_{n-1}^i - \Omega_n(i) \boldsymbol{v}_n^i (\boldsymbol{v}_n^i)^T / \beta_n^i$,
9:     $\boldsymbol{u}_n^i = \boldsymbol{u}_{n-1}^i + \Omega_n(i) \left[\boldsymbol{x}_n(i) - \boldsymbol{w}_n^T \boldsymbol{u}_{n-1}^i \right] \boldsymbol{R}_n^i \boldsymbol{w}_n$.
10:   **end for**
11:   $n := n + 1$;
12: **until** termination

has been proven to be equivalent to the GROUSE gradient algorithm for a given step size [53], showing that indeed the distinction of "algebraic" and "geometric" algorithms is not fundamental.

Remark 2 (SNIPE). *A block variant of GROUSE was presented in [63], called Subspace Navigation via Interpolation from Partial Entries (SNIPE). This algorithm partitions the input into blocks and for each block optimizes a subspace to fit the observed entries on that block but remain close to the previous subspace estimate.*

B. PAST

The Projection Approximation Subspace Tracking (PAST) is proposed by Yang [25], [28] for subspace tracking with full data, which is described in Algorithm 5. PAST optimizes the following function at time n without constraining $\boldsymbol{U}$ to have orthogonal columns:

$$\boldsymbol{U}_n = \operatorname*{argmin}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \|\boldsymbol{x}_\ell - \boldsymbol{U} \boldsymbol{U}^T \boldsymbol{x}_\ell\|_2^2, \quad (24)$$

where prior observations are discounted by a geometric factor $0 \ll \lambda \leq 1$. The name "projection approximation" comes from the fact that the projection onto the subspace $\boldsymbol{U}$ is approximated by $\boldsymbol{U} \boldsymbol{U}^T$, without the constraint $\boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{I}_k$. This sum is further approximated by replacing the second $\boldsymbol{U}$ in (24) by $\boldsymbol{U}_{\ell-1}$, yielding

$$\boldsymbol{U}_n = \operatorname*{argmin}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \|\boldsymbol{x}_\ell - \boldsymbol{U} \boldsymbol{U}_{\ell-1}^T \boldsymbol{x}_\ell\|_2^2. \quad (25)$$

Let the coefficient vector be $\boldsymbol{w}_\ell = \boldsymbol{U}_{\ell-1}^T \boldsymbol{x}_\ell$, then (25) can be rewritten as

$$\boldsymbol{U}_n = \operatorname*{argmin}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \|\boldsymbol{x}_\ell - \boldsymbol{U} \boldsymbol{w}_\ell\|_2^2, \quad (26)$$

whose solution can be written in a closed-form and efficiently found via recursive least-squares. The PAST algorithm has a computational complexity of $\mathcal{O}(dk)$. PAST has been very popular due to its efficiency, and it has been extended and modified in various ways [20], [64], [65].

C. PETRELS

The PETRELS algorithm, proposed in [23], can be viewed as a modification of the PAST algorithm to handle missing data, which is summarized by Algorithm 6. PETRELS optimizes the following function at time n without constraining $\boldsymbol{U}$ to have orthogonal columns:

$$\boldsymbol{U}_n = \operatorname*{argmin}_{\boldsymbol{U} \in \mathbb{R}^{d \times k}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \min_{\boldsymbol{w}_\ell \in \mathbb{R}^k} \|\mathcal{P}_{\Omega_\ell}(\boldsymbol{x}_\ell - \boldsymbol{U} \boldsymbol{w}_\ell)\|_2^2. \quad (27)$$

At each time n, PETRELS alternates between coefficient estimation and subspace update. We first estimate the coefficient vector by minimizing the projection residual using the previous subspace estimate:

$$\boldsymbol{w}_n = \operatorname*{argmin}_{\boldsymbol{w}} \|\mathcal{P}_{\Omega_n} (\boldsymbol{x}_n - \boldsymbol{U}_{n-1} \boldsymbol{w})\|_2^2, \quad (28)$$

where $\boldsymbol{U}_0 \in \mathbb{R}^{d \times k}$ is a random subspace initialization. The subspace $\boldsymbol{U}$ is then updated by minimizing

$$\boldsymbol{U}_n = \operatorname*{argmin}_{\boldsymbol{U}} \sum_{\ell=1}^{n} \lambda^{n-\ell} \|\mathcal{P}_{\Omega_\ell} (\boldsymbol{x}_\ell - \boldsymbol{U} \boldsymbol{w}_\ell)\|_2^2, \quad (29)$$

where $\boldsymbol{w}_\ell, \ell = 1, \cdots, n$ are estimates from (28). The objective function in (29) decomposes into a parallel set of smaller problems, one for each row of $\boldsymbol{U}_n = [\boldsymbol{u}_n^1, \boldsymbol{u}_n^2, \cdots, \boldsymbol{u}_n^d]^T$, where $\boldsymbol{u}_n^i \in \mathbb{R}^k$. Thus the ith row can be estimated by solving

$$\boldsymbol{u}_n^i = \operatorname*{argmin}_{\boldsymbol{u} \in \mathbb{R}^k} \sum_{\ell=1}^{n} \lambda^{n-\ell} \Omega_\ell(i)(\boldsymbol{x}_\ell(i) - \boldsymbol{w}_\ell^T \boldsymbol{u})^2$$
$$= \boldsymbol{u}_{n-1}^i + \Omega_\ell(i) \cdot \left[\boldsymbol{x}_n(i) - \boldsymbol{w}_n^T \boldsymbol{u}_{n-1}^i \right]$$
$$\cdot \left(\sum_{\ell=1}^{n} \lambda^{n-\ell} \Omega_\ell(i) \boldsymbol{w}_\ell \boldsymbol{w}_\ell^T \right)^{-1} \boldsymbol{w}_n \quad (30)$$

for $i = 1, \cdots, d$. Again, the problem can be solved efficiently via recursive least-squares. Moreover, PETRELS can be made very efficient by parallelizing the implementation of (30).

Both PAST and PETRELS can be regarded as applying second-order stochastic gradient descent [57] to the loss function, and each step of the update is approximately a Newton step. Therefore, it is expected that the algorithm will converge quadratically when it is close to the optimal solution. Several

algorithms can be developed along similar lines of PETRELS, where the loss function is revised to include regularization terms on the Frobenius norms of the subspace $\boldsymbol{U}$ and the weight vector $\boldsymbol{w}_n$, which we refer the readers to [26], [27].

V. PERFORMANCE ANALYSIS

In this section we will describe general analysis methodologies as well as specific theoretical results for characterizing the performance of the aforementioned streaming PCA and subspace tracking algorithms.

To carry out the analysis, we need to make assumptions on how the data are generated. A popular approach that has been taken in the literature is to assume that each data vector is generated according to the following "spiked model" [66]:

$$\boldsymbol{x}_n = \boldsymbol{U}^* \boldsymbol{a}_n + \sigma \boldsymbol{\epsilon}_n, \tag{31}$$

where $\boldsymbol{U}^*$ is a deterministic $d \times k$ orthogonal matrix, $\boldsymbol{a}_n$ is a random signal vector with covariance matrix $\boldsymbol{\Sigma}_a$, and $\boldsymbol{\epsilon}_n$ is the noise vector. For simplicity, we assume that the covariance matrix of $\boldsymbol{\epsilon}_n$ is the identity matrix $\boldsymbol{I}_d$, and we use σ to denote the noise level. This model arises in applications such as array signal processing, where $\boldsymbol{U}^*$ is the "steering matrix" and k denotes the number of targets to be tracked by the array. The generative model (31) can also be seen as a special case of the probabilistic model described in Section II-A, since $\mathbb{E}[\boldsymbol{x}_n \boldsymbol{x}_n^T] = \boldsymbol{U}^* \boldsymbol{\Sigma}_a \boldsymbol{U}^{*T} + \sigma^2 \boldsymbol{I}_d$ is a sum of an exact low-rank matrix and a full-rank identity matrix. In the missing data case studied in this paper, only a subset of the coordinates of $\boldsymbol{x}_n$ are observed. Thus, the actual measurement is $\boldsymbol{y}_n = \mathcal{P}_{\Omega_n}(\boldsymbol{x}_n)$ as in (5), where $\mathcal{P}_{\Omega_n}$ denotes the projection operator onto an observation mask, $\Omega_n \in \{0,1\}^d$. The ith entry of $\boldsymbol{x}_n$ is observed if and only if $\Omega_n(i) = 1$. For simplicity, we shall assume that $\{\Omega_n(i)\}$ is a collection of i.i.d. binary random variables such that

$$\mathbb{P}(\Omega_n(i) = 1) = \alpha, \tag{32}$$

for some constant $\alpha \in (0, 1)$.

A. Classical Asymptotic Analysis

Historically, the first analysis of subspace tracking algorithms was done in the asymptotic regime (see, *e.g.*, [28], [29]), where the algorithms are shown to converge, in the small step size limit, to the solution of some deterministic Ordinary Differential Equations (ODEs).

To understand the basic ideas underlying such analysis, we note that the essence of almost all the online algorithms described in Section III and Section IV is a stochastic recursion of the form

$$\boldsymbol{U}_n = \boldsymbol{U}_{n-1} + \eta_n Q(\boldsymbol{U}_{n-1}, \boldsymbol{x}_n, \Omega_n). \tag{33}$$

Here, $\boldsymbol{U}_n$ is the estimate at time n; $Q(\cdot, \cdot, \cdot)$ is some nonlinear function of the previous estimate $\boldsymbol{U}_{n-1}$, the new complete data vector $\boldsymbol{x}_n$, and its observation mask Ω_n; and η_n is the step size (*i.e.*, the learning rate). For example, Krasulina's method given in (19) is just a special case of (33) [with $\Omega_n(i) \equiv 1$]. When the step size η_n is small, we can perform Taylor's expansion (with respect to η_n) on the recursion formulas of Oja's method

(16) and GROUSE (23), and show that these two algorithms can also be written in the form of (33) after omitting higher-order terms in η_n.

Under the statistical model (31) and (32), the general algorithm (33) is simply a Markov chain with state vectors $\boldsymbol{U}_n \in \mathbb{R}^{d \times k}$. The challenge in analyzing the convergence of (33) comes from the nonlinearity in the function $Q(\cdot, \cdot, \cdot)$. In the literature, a very powerful analytical tool is the so-called ODE method. It was introduced to the control and signal processing communities by Ljung [30] and Kushner [31] in the 1970s, and similar approaches have an even longer history in the literature of statistical physics and stochastic processes (see, *e.g.*, [67], [68] for some historical remarks).

The basic idea of the ODE method is to associate the discrete-time stochastic process (33) with a continuous-time *deterministic* ODE. Asymptotically, as the step size $\eta_n \to 0$ and the number of steps $n \to \infty$, the process (33) can be shown to converge to the solution of an ODE. Specifically, we let the step sizes be such that

$$\sum_{n=1}^{\infty} \eta_n = \infty \text{ and } \sum_{n=1}^{\infty} \eta_n^2 < \infty.$$

For example, a popular choice is $\eta_n = c/n$ for some $c > 0$. By defining $t_n = \sum_{\ell \le n} \eta_\ell$ as the "fictitious" time, we can convert the discrete-time process $\boldsymbol{U}_n$ to a continuous-time process $\boldsymbol{U}_t$ via linear interpolation:

$$\boldsymbol{U}_t = \boldsymbol{U}_{n-1} + \frac{t - t_{n-1}}{t_n - t_{n-1}} (\boldsymbol{U}_n - \boldsymbol{U}_{n-1}), \ t_{n-1} \le t \le t_n. \tag{34}$$

Under certain regularity conditions on the function $Q(\cdot, \cdot, \cdot)$, one can then show that, as $t \to \infty$, the randomness in the trajectory of $\boldsymbol{U}_t$ will diminish and $\boldsymbol{U}_t$ will converge to the deterministic solution of an ODE [30], [31].

Although a rigorous proof of the above convergence is technical, the limiting ODE, if the convergence indeed holds, can be easily derived, at least in a non-rigorous way. To start, we can rewrite (33) as

$$\frac{\boldsymbol{U}_n - \boldsymbol{U}_{n-1}}{\eta_n} = \mathbb{E}_{\boldsymbol{x}_n, \Omega_n | \boldsymbol{U}_{n-1}}[Q(\boldsymbol{U}_{n-1}, \boldsymbol{x}_n, \Omega_n)] + m_n, \tag{35}$$

where $\mathbb{E}_{\boldsymbol{x}_n, \Omega_n | \boldsymbol{U}_{n-1}}[\cdot]$ denotes the conditional expectation of the "new information" $\boldsymbol{x}_n, \Omega_n$ given the current state $\boldsymbol{U}_{n-1}$, and m_n captures the remainder terms. From the construction of $\boldsymbol{U}_t$ in (34), the left-hand side of (35) is equal to $(\boldsymbol{U}_{t_{n-1}+\eta_n} - \boldsymbol{U}_{t_{n-1}})/\eta_n$, which converges to $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{U}_t$ since the step size $\eta_n \to 0$. Moreover, one can show that the remainder m_n is of order $o(1)$. It follows that we can write the limit form of (35) as an ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{U}_t = h(\boldsymbol{U}_t), \tag{36}$$

where $h(\boldsymbol{U}_t) = \mathbb{E}_{\boldsymbol{x}, \Omega \,|\, \boldsymbol{U}}[Q(\boldsymbol{U}, \boldsymbol{x}, \Omega)]$.

The ODE approach is a very powerful analysis tool. By studying the fixed points of the limiting dynamical system in (36), we can then draw conclusions about the convergence behavior of the original stochastic process (33). This approach was taken in [28], where the author used an ODE analysis to show that the PAST algorithm [25] globally converges to the target signal subspace $\boldsymbol{U}^*$ with probability one. This result

was later adapted in [23] to analyze PETRELS for the fully observed case.

B. Asymptotic Analysis in the High-Dimensional Regime

Despite its versatility and strong theoretical value, the above classical asymptotic approach has several limitations: First, the analysis requires the step size η_n to tend to zero as $n \to \infty$. While using a decreasing sequence of step sizes η_n helps the stochastic algorithm to converge to the globally optimal solution, it is not a good strategy for applications where the target low-dimensional subspace can be time-varying. In that scenario, a small but fixed step size is often more preferable, as it would make the algorithms more nimble in tracking the changing subspace. Second, the classical asymptotic analysis leads to an ODE with $\mathcal{O}(d)$ variables. In modern applications, the number of variables, *i.e.*, d can be very large, making it less practical to numerically solve the ODE.

In what follows, we briefly review a different asymptotic analysis approach [32]–[35] that addresses the above problems. For simplicity, we present the underlying idea using the example of Oja's method (16) for learning a one-dimensional subspace using full data, although the same approach applies to the general rank-k case with missing data.

When $k = 1$, the orthogonalization operator Π in (16) is just a normalization, and thus the update rule can be simplified as

$$\boldsymbol{U}_n = \frac{\boldsymbol{U}_{n-1} + \eta_n \boldsymbol{x}_n \boldsymbol{x}_n^T \boldsymbol{U}_{n-1}}{\left\| \boldsymbol{U}_{n-1} + \eta_n \boldsymbol{x}_n \boldsymbol{x}_n^T \boldsymbol{U}_{n-1} \right\|}. \tag{37}$$

This stochastic process is a Markov chain in $\mathbb{R}^d$, where the dimension d can be large. To reduce the underlying dimension of the system we need to analyze, we note that the quality of the estimate $\boldsymbol{U}_n$ can be fully captured by a scalar quantity

$$s_n \overset{\text{def}}{=} \frac{\boldsymbol{U}_n^T \boldsymbol{U}^*}{\|\boldsymbol{U}_n\| \|\boldsymbol{U}^*\|}. \tag{38}$$

Clearly, $s_n \in [-1, 1]$, with $s_n = \pm 1$ indicating perfect alignment of $\boldsymbol{U}^*$ and $\boldsymbol{U}_n$. In what follows, we refer to s_n as the *cosine similarity*.

Substituting (37) and (31) into (38), we get a recursion formula for the cosine similarity:

$$s_n = \frac{s_{n-1} + \eta_n (a_n + \sigma p_n)(a_n s_{n-1} + \sigma q_n)}{\left(1 + \eta_n (a_n s_{n-1} + \sigma q_n)^2 [2 + \eta_n (a_n^2 + \sigma^2 \|\boldsymbol{\epsilon}_n\|^2 + 2\sigma a_n p_n)] \right)^{\frac{1}{2}}}, \tag{39}$$

where $a_n, \boldsymbol{\epsilon}_n$ are the signal and noise vector in the generating model (31), respectively, and $p_n \overset{\text{def}}{=} \boldsymbol{\epsilon}_n^T \boldsymbol{U}^*$ and $q_n \overset{\text{def}}{=} \boldsymbol{\epsilon}_n^T \boldsymbol{U}_{n-1}$. The expression (39) might appear a bit complicated, but the key observation is the following: If the noise vector $\boldsymbol{\epsilon}_n$ is drawn from the normal distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, then it follows from the rotational symmetry of the multivariate normal distribution that

$$\mathbb{P}(p_n, q_n | s_{n-1}) \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & s_{n-1} \\ s_{n-1} & 1 \end{bmatrix} \right).$$

In other words, given s_{n-1}, the two random variables p_n and q_n are joint normal random variables whose distribution is a function of s_{n-1}. Consequently, the recursion (39) from s_{n-1} to s_n forms a one-dimensional Markov chain. Note that this exact Markovian property relies on the assumption that the noise vector $\boldsymbol{\epsilon}_n$ be normally distributed. However, due to the central limit theorem, we can show that this property still holds asymptotically, when the underlying dimension d is large and when the elements of $\boldsymbol{\epsilon}_n$ are independently drawn from more general distributions with bounded moments [35]. Moreover, these arguments can be generalized to the missing data case, provided that the subsampling process follows the probabilistic model in (32).

Further analysis shows that, by choosing the step size $\eta_n = \tau/d$ for some fixed $\tau > 0$, we can apply the similar ODE idea used in the classical asymptotic analysis to obtain a deterministic, limit process for the cosine similarity. More specifically, we can show that a properly time-rescaled version of $s^{(d)}(t) \overset{\text{def}}{=} s_{\lfloor td \rfloor}$ will converge weakly, as $d \to \infty$, to a deterministic function that is characterized as the unique solution of an ODE (see [32]–[35] for details).

In [33], [35], the exact dynamic performance of Oja's method, GROUSE and PETRELS was analyzed in this asymptotic setting. In what follows, we only state the results for $k = 1$. For simplicity, we also assume that the covariance matrix of the signal vector $\boldsymbol{a}_n$ in (31) is $\boldsymbol{\Sigma}_a = \boldsymbol{I}_k$. For PETRELS, it turns out that we just need to study two scalar processes: the cosine similarity s_n as defined in (38) and an auxiliary parameter

$$g_n = d R_n \|\boldsymbol{U}_n\|^{-2},$$

where R_n is the average of the quantities $\{R_n^i\}$ in Algorithm 6. Accordingly, the parameter δ in the algorithm also needs to be rescaled such that $\delta = \delta'/d$ for some fixed $\delta' > 0$. By introducing the "fictitious" time $t = n/d$, we can embed the discrete-time sequences s_n, g_n into continuous-time as $s^{(d)}(t) = s_{\lfloor td \rfloor}$ and $g^{(d)}(t) = g_{\lfloor td \rfloor}$. As the underlying dimension $d \to \infty$, we can show that the stochastic processes $\left\{ s^{(d)}(t), g^{(d)}(t) \right\}$ converge weakly to the unique solution of the following systems of coupled ODEs:

$$\begin{aligned} \frac{\mathrm{d}s(t)}{\mathrm{d}t} &= \alpha s(1 - s^2)g - \frac{\sigma^2}{2}(\alpha s^2 + \sigma^2)sg^2 \\ \frac{\mathrm{d}g(t)}{\mathrm{d}t} &= -g^2(\sigma^2 g + 1)(\alpha s^2 + \sigma^2) + \mu g, \end{aligned} \tag{40}$$

where α is the probability with which each coordinate of the data vectors can be observed [see (32)], and $\mu > 0$ is a constant such that the discount parameter λ in (27) is set to $\lambda = 1 - \mu/d$.

Compared to the classical ODE analysis [23], [28] which keeps the ambient dimension d *fixed* and studies the asymptotic limit as the step size tends to 0, the ODEs in (40) only involve 2 variables $s(t)$ and $g(t)$. This low-dimensional characterization makes the new limiting results more practical to use, especially when the dimension is large.

Similar asymptotic analysis can also be carried out for Oja's method and GROUSE (see [35]). Interestingly, the time-varying cosine similarities s_n associated with the two algo-
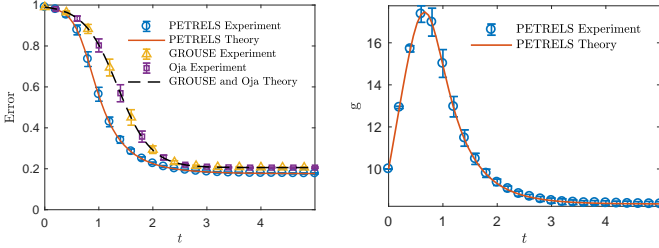
Fig. 1: Monte Carlo simulations of the Oja's method, GROUSE and PETRELS v.s. asymptotic predictions obtained by the limiting ODEs given in (40) and (41). The error is defined as $1 - s^2(t)$. The signal dimension is $d = 10^4$, the noise parameter is $\sigma = 0.2$, and the subsampling probability is $\alpha = 0.17$. The error bars shown in the figure correspond to one standard deviation over 50 independent trials. The simulation results also confirm the prediction that Oja's method and GROUSE converge to the same deterministic limit.
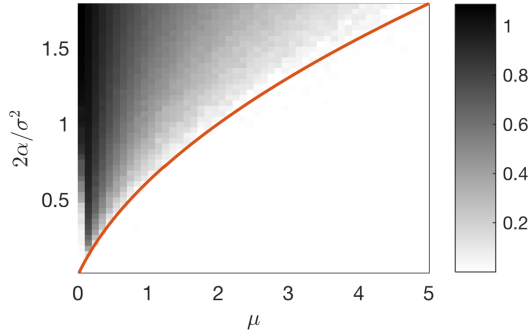
Fig. 2: The grayscale in the figure visualizes the steady-state errors of the PETRELS algorithm corresponding to different values of the noise variance σ^2, the subsampling ratio α, and the discount parameter μ. The red curve is the theoretical prediction given in (42) of a phase transition boundary, below which no informative solution can be achieved by the algorithm. The theoretical prediction matches well with numerical results.

rithms are asymptotically equivalent, with both converging, as $d \to \infty$, to the solution of a limiting ODE:

$$\frac{\mathrm{d}s}{\mathrm{d}t} = \tau\big(\alpha - \tfrac{\tau\sigma^4}{2}\big)s - \alpha\tau\big(1 + \tfrac{\tau\sigma^2}{2}\big)s^3, \qquad (41)$$

where $\tau > 0$ is a constant such that the step size parameter η_n used in Algorithms 3 and 4 is $\eta_n = \tau/d$, and α is again the subsampling probability. Numerical verifications of the asymptotic results are shown in Figure 1. We can see that the theoretical prediction given by the ODEs (40) and (41) can accurately characterize the actual dynamic performance of the three algorithms.

The convergence behavior of the algorithms can also be established by analyzing the fixed points of the dynamical system associated with the limiting ODEs. For example, by studying the stability of the fixed points of (40) for PETRELS (see [35]), one can show that $\lim_{t\to\infty} s(t) > 0$ if only if

$$\mu < \Big(2\alpha/\sigma^2 + 1/2\Big)^2 - 1/4, \qquad (42)$$

where $\alpha \in (0, 1)$ is the subsampling probability and $\sigma > 0$ is the noise level in (31). A "noninformative" solution corresponds to $s(t) = 0$, in which case the estimate $\boldsymbol{U}_n$ and the underlying subspace $\boldsymbol{U}^*$ are orthogonal (i.e., uncorrelated). The expression in (42) predicts a phase transition phenomenon for PETRELS, where a critical choice of μ (as a function of α and σ) separates informative solutions from non-informative ones. This prediction is confirmed numerically in Figure 2.

C. Finite Sample Analysis

In addition to the asymptotic analysis described in the previous subsections, there have also been many recent efforts in establishing finite-sample performance guarantees for various streaming PCA algorithms. We begin with analysis in the case of fully observed data vectors.

One of the earlier works is [36], where the authors analyze a block variant of Oja's method: within each iteration, multiple sample vectors are drawn, whose empirical covariance matrix is then used in place of $\boldsymbol{x}_n\boldsymbol{x}_n^T$ in (37). Under the generative model (31), the authors show that this algorithm can reach accuracy $\|\boldsymbol{U}_n - \boldsymbol{U}^*\| \le \varepsilon$, in the rank-one case, if the total number of samples is of order

$$n = \mathcal{O}\left(\frac{(1 + 3(\sigma + \sigma^2)\sqrt{d})^2 \log(d/\varepsilon)}{\varepsilon^2 \log[(\sigma^2 + 3/4)/(\sigma^2 + 1/2)]}\right),$$

where σ is the noise level in (31). Similar analysis is available for the general rank-k case. Block Oja's methods have also been studied in [37], [56], but the analysis is done under a model much more general than (31): the data vectors $\boldsymbol{x}_n$ are assumed to be drawn i.i.d. from a general distribution on $\mathbb{R}^d$ with zero-mean and covariance $\boldsymbol{\Sigma}$.

The performance of Oja's original method has also been studied under the above general model. For the rank-1 case, Li *et al.* [38] established that the error is on the order of $\mathcal{O}\left(\frac{\lambda_1\lambda_2}{(\lambda_1 - \lambda_2)^2}\frac{d\log n}{n}\right)$, which is near optimal up to a logarithmic factor, with the step size $\eta_n = \frac{2\log n}{(\lambda_1 - \lambda_2)n}$. Similarly, Jain *et al.* [69] provide a near-optimal result for Oja's method, with a time-varying learning rate $\eta_n = \frac{1}{(\lambda_1 - \lambda_2)(n + n_1)}$, where n_1 is some starting time. Other results include Balsubramani *et al.* [18], Shamir [39], Li *et al.* [70]. Allen-Zhu and Li [40] provide a near-optimal guarantee for rank-k Oja's method very recently. This paper also contains a comprehensive table that summarizes many recent results.

Most of the existing analysis and performance guarantees in the literature assume that there is a positive gap between λ_k and λ_{k+1}. This eigengap assumption was removed in Shamir [39] and Allen-Zhu and Li [40], where the authors provide sample complexity bounds that are gap-free, *i.e.* they do not require a strictly positive eigengap.

For fully observed data vectors, the global convergence of the GROUSE algorithm was established in [59], [60] under the generative model (31) for the special noise-free case, *i.e.*, $\sigma = 0$. Define $\zeta_n := \det\big(\boldsymbol{U}^{*T}\boldsymbol{U}_n\boldsymbol{U}_n^T\boldsymbol{U}^*\big) \in [0, 1]$ to be the

"determinant similarity[6]" between two subspaces, which will be 0 when the subspaces have any orthogonal direction and will be 1 if and only if the two subspaces are equal. Let $\zeta^* < 1$ be the desired accuracy of our estimated subspace. Initialize the starting point (U_0) of GROUSE as the orthonormalization of a $d \times k$ matrix with entries being standard normal variables. Then for any $\rho > 0$, after

$$n \geq \underbrace{\left(\frac{2k^2}{\rho} + 1\right) \mu_0 \log(d)}_{n_1} + \underbrace{2k \log\left(\frac{1}{2\rho(1 - \zeta^*)}\right)}_{n_2}$$

iterations of GROUSE Algorithm 4, $\zeta_n \geq \zeta^*$ with probability at least $1 - 2\rho$, where $\mu_0 = 1 + \frac{\log \frac{(1-\rho')}{C} + k \log(e/k)}{k \log d}$ with $C > 0$ a constant approximately equal to 1. This result is divided into two parts: n_1 is the bound on the number of observed vectors (also iterations) required to get to a basin of attraction, and n_2 is the number required for linear convergence in the basin of attraction. In practice we see that the bound on n_1 is loose but n_2 is accurate when discarding dependence on ρ (see [59], [60] for more details.)

In contrast to the fully observed case, the literature is much sparser for finite-sample analysis in the missing data case. While a great deal of work has established performance guarantees in the batch setting (*i.e.* the low-rank matrix completion problem [7], [71]–[74]), the only results in the streaming case with missing data are local convergence results for GROUSE and the work of [75], where the authors consider recovering a low-rank positive semidefinite matrix observed with missing data using stochastic gradient descent; while the authors of [75] did not study streaming PCA in particular, their method could be applied in the streaming data case.

Bounds on the expected improvement at each iteration for the GROUSE algorithm has been given for the case of noisy data [59] and missing or compressed data [58], [60]. Under some assumptions of data regularity and subspace incoherence, these convergence results show that the GROUSE algorithm improves ζ_{n+1} in expectation as follows:

$$\mathbb{E}\left[\zeta_{n+1} | U_n\right] \geq \left(1 + \eta \frac{|\Omega_n|}{d} \frac{1 - \zeta_n}{k}\right) \zeta_n \qquad (43)$$

with high probability, where $\eta \approx 1$ is slightly different for each sampling type [59], [60]. The expectation is taken with respect to the generative model (31) and the subsampling model (32). These results for missing or compressed data can be generalized to a local convergence result [58], [60].

In the work of [75], the authors propose a step size scheme for general Euclidean stochastic gradient descent with which they prove global convergence results from a randomly generated initialization. Their choice of step size depends on the knowledge of some parameters that are likely to be unknown in practical problems. Without this knowledge, the results only hold with sufficiently small step size that implies very slow convergence.

---

[6]Note that the rank-k generalization of the aforementioned cosine similarity (38) is given as $\|U_n^T U^*\|_F^2$ for U_n and U^* with orthonormal columns. Therefore the two similarity measures are related via the singular values of $U_n^T U^*$.

It remains an important open problem to establish finite sample global performance guarantees in the missing data case for GROUSE and other algorithms such as Oja's and PETRELS.

VI. NUMERICAL EXPERIMENTS

We benchmark the performance of several competitive algorithms reviewed in this paper that are able to handle missing data, including GROUSE [22], PETRELS [23], PIMC [54], MD-ISVD [3], Brand's algorithm [15] and Oja's algorithm [17] adapted to the missing data case as proposed in this paper in Algorithm 3. The code for these simulations is available in Python Jupyter notebooks at http://gitlab.eecs.umich.edu/girasole/ProcIEEEstreamingPCA/.

A. Simulation Setup

Let $d = 200$ and $k = 10$. We select the ground truth subspace as $U^* = \text{orth}(\widetilde{U}) \in \mathbb{R}^{d \times k}$ where $\widetilde{U}$ is composed of standard i.i.d. Gaussian entries. The coefficient vectors are generated as $a_n \sim \mathcal{N}(0, \text{diag}(c))$, where the loading vector c is given as

$$c = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]^T,$$

for a well-conditioned system, and

$$c = [1, 1, 1, 1, 1, 0.3, 0.3, 0.3, 0.1, 0.1]^T$$

for an ill-conditioned system. The data vector is generated using (31), where ϵ_n is composed of i.i.d. $\mathcal{N}(0, \sigma^2)$ entries, with the noise level given as $\sigma = [10^{-2}, 10^{-5}, 0]$. Each vector is observed with a fixed percent $\alpha = [0.1, 0.5, 1]$ of entries selected uniformly at random. We note that the lowest sampling fraction is near the information-theoretic lower bound of the number of uniform random samples that will guarantee matrix reconstruction. The reconstruction error is calculated as the projection error $\|(I - P_{\widehat{U}})U^*\|_F^2$, where $\widehat{U}$ is the estimated (orthogonalized) subspace. All the algorithms are initialized with the same orthogonalized random subspace. We assume all algorithms are given the true rank. Throughout the simulations, we set the discount parameter $\eta = 0.98$ in PETRELS and $\beta = 0.98$ in Brand's algorithm, and set $\eta = 0.5$ in Oja's algorithm. Note that we did a grid search for these parameters using the setting $\sigma = 10^{-5}, \alpha = 0.5$, and so picking them differently will likely yield different trade-offs in performance for other levels of noise and missing data. Our goal is to illustrate their typical behaviors without claiming relative superiority.

B. Performance Evaluations

We first examine the performance of subspace estimation with streaming observations of a static subspace. Fig. 3 shows the reconstruction error of the algorithms with respect to the number of snapshots for both the well-conditioned case and the ill-conditioned case. For each algorithm, the dark line is the median of 50 runs, and the transparent ribbon shows the 25% and 75% quantiles. Note how in most algorithms, the ribbon is nearly invisible, meaning that the algorithm execution with different initialization, generative subspace, random data
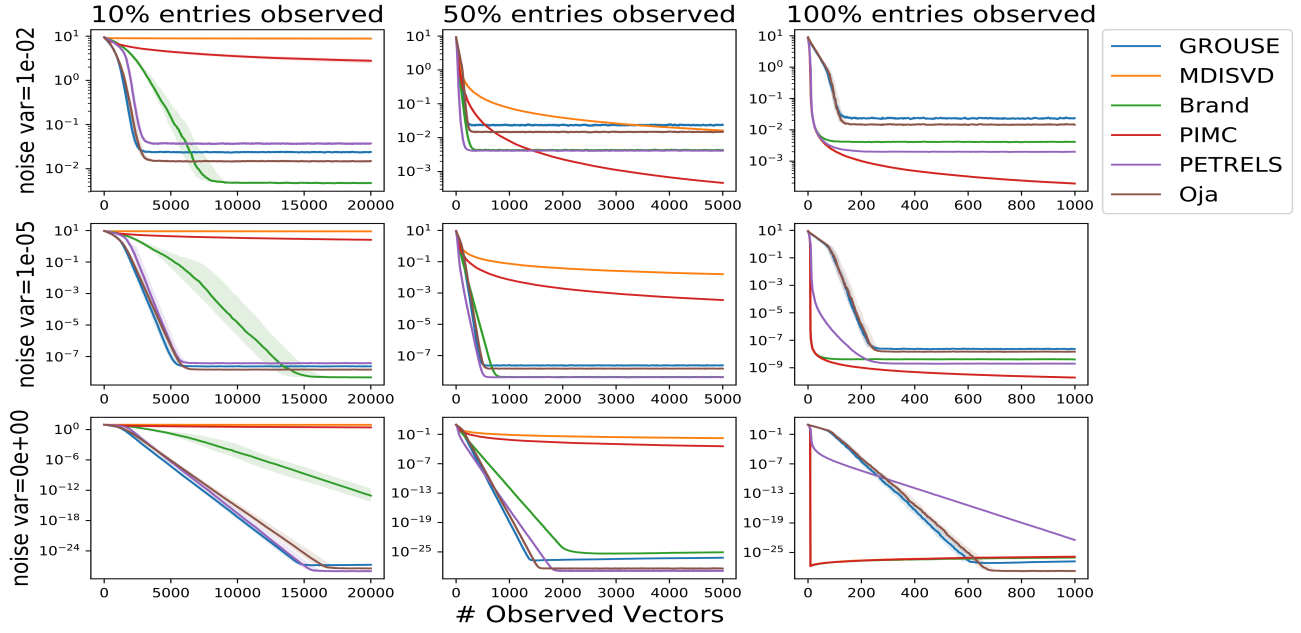
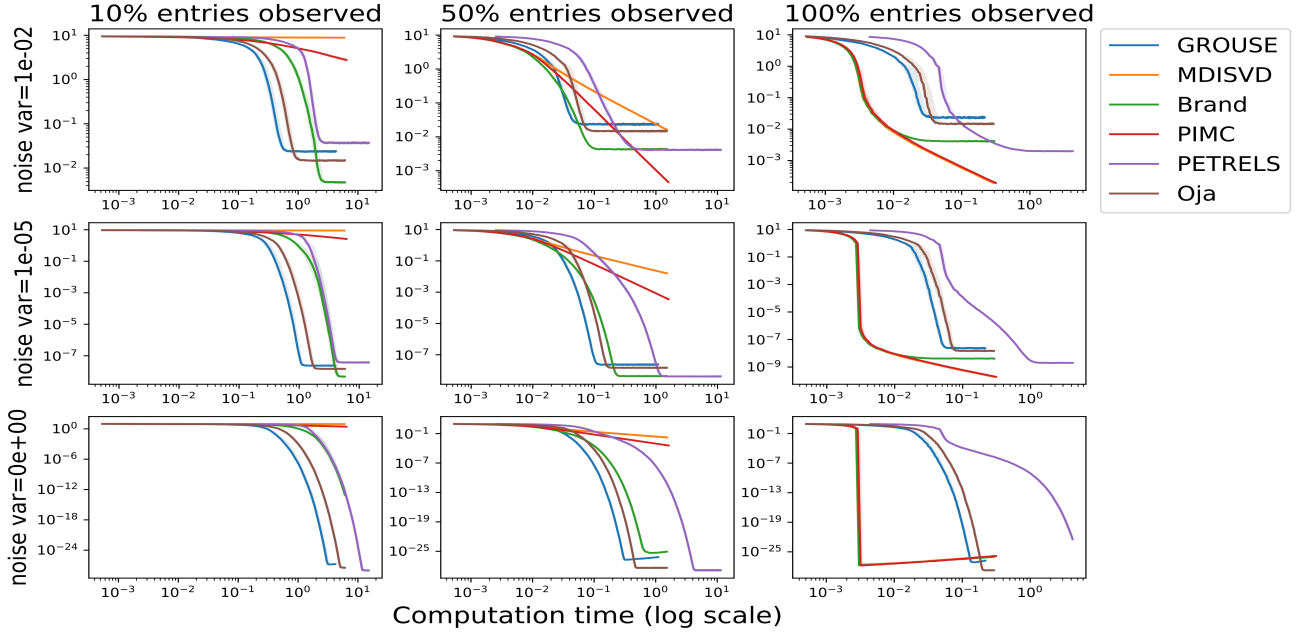(a) Subspace error versus number of observed vectors in the well-conditioned case
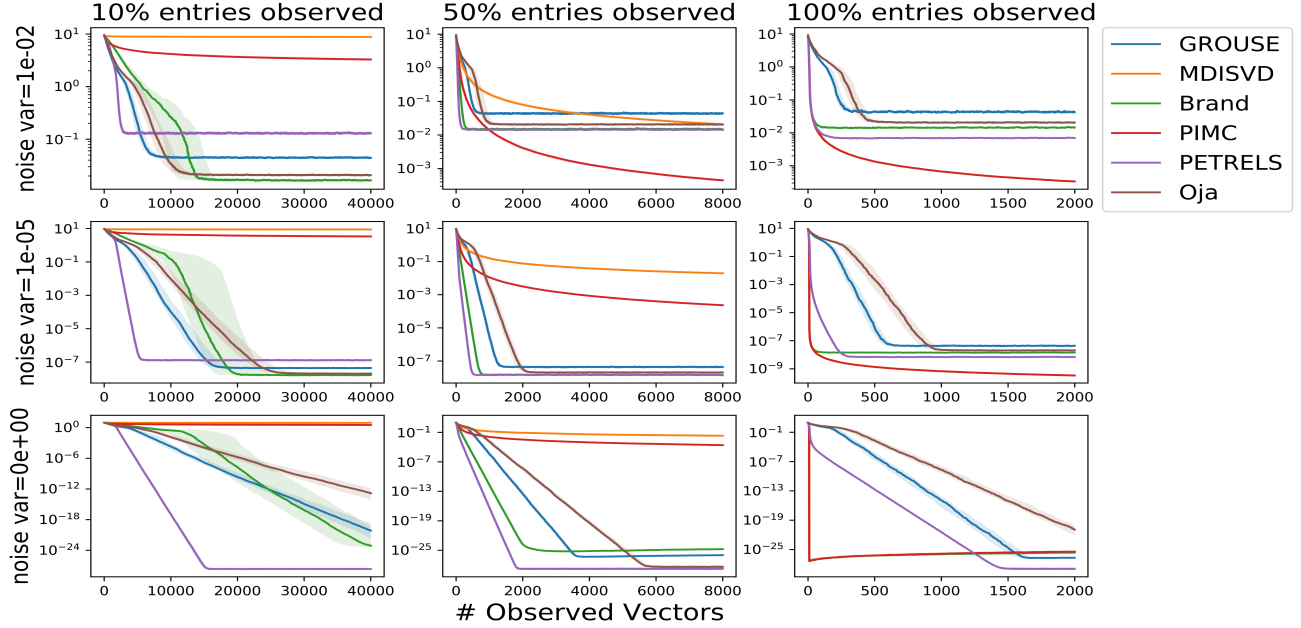
(b) Subspace error versus number of observed vectors in the ill-conditioned case

Fig. 3: Reconstruction error versus the number of observed snapshots for various algorithms in both well-conditioned and ill-conditioned cases for streaming PCA where every data vector is observed once.

draw from that subspace, and random missing data patterns have highly consistent behavior, especially as the number of observed entries gets larger. All of the examined algorithms are capable of estimating the subspace after seeing sufficiently many data vectors. It can be seen that as the fraction of missing data increases, more snapshots are needed for the algorithms to converge. Also as the noise level increases, the algorithms converge to a higher error level.

We next show the same plots but as a function of computation time, i.e. assuming that the streaming vectors are available as soon as each algorithm finishes processing the

previous vector. Fig. 4 shows the reconstruction error of the algorithms with respect to the wall-clock time for both the well-conditioned case and the ill-conditioned case. For each algorithm, we take the median computation time at each iteration over 50 runs to get values on the x-axis. Then the the dark line is the median error of 50 runs interpolated to those computation times. Again the transparent ribbon shows the 25% and 75% error quantiles. We show the time on a log axis since different algorithms take different orders of magnitude computation time. Note that PETRELS can be accelerated by computing in parallel the updates for each row of the subspace,

(a) Subspace error versus computation time for the well-conditioned case
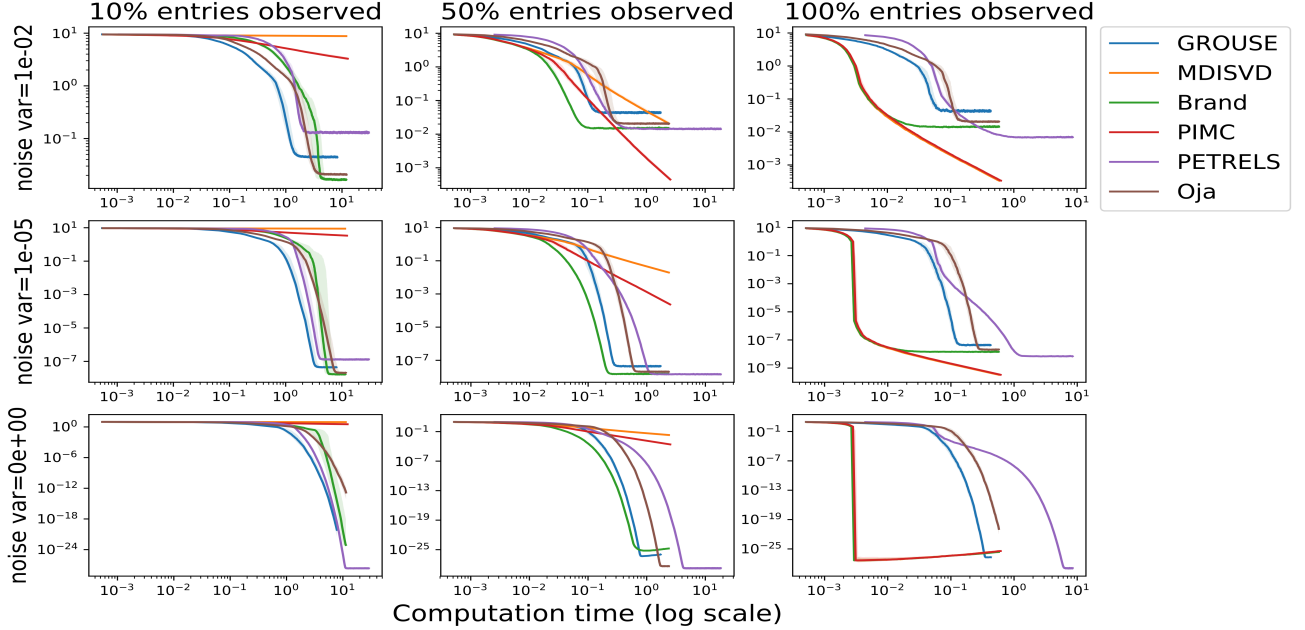
(b) Subspace error versus computation time for the ill-conditioned case

Fig. 4: Reconstruction error versus the wall-clock time for various algorithms in both well-conditioned and ill-conditioned cases for one-pass streaming PCA.

which is not implemented here. Again, while many algorithms converge very quickly with complete data, they may take significantly longer in the presence of missing data.

Next, we examine the performance of the algorithms on subspace tracking, where we assume there is an abrupt change in the underlying subspace and the goal is to examine how fast the algorithms are able to track the change. Moreover, we generate the loading vector c before and after the same with entries drawn from a uniform distribution in $[0, 1]$. The noise level is set as $\sigma = 10^{-5}$ and the fraction of observation is set as $\alpha = 0.3$. Fig. 5 shows the performance of the algorithms when

the subspace changes abruptly at the 4000th snapshot. Note that MD-ISVD and PIMC have poor performance after the subspace change; these algorithms seem to rely more heavily on historical data through their singular value estimates.

Finally in a second experiment for subspace tracking, we assume the underlying subspace is slowly rotating over time. We achieve this by initializing a random subspace with orthonormal columns $U_0 \in \mathbb{R}^{d \times k}$ and sample a skew-symmetric matrix $B \in \mathbb{R}^{d \times d}$ with independent, normally distributed entries. We set $U_t = \exp(\delta_0 B) U_{t-1}$ for some small rotation parameter δ_0 and where $\exp$ is the matrix exponential. In our
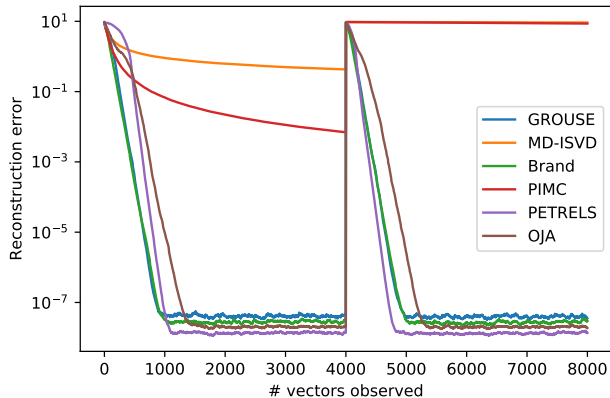
Fig. 5: Reconstruction error versus the number of observed snapshots with an abrupt subspace change.
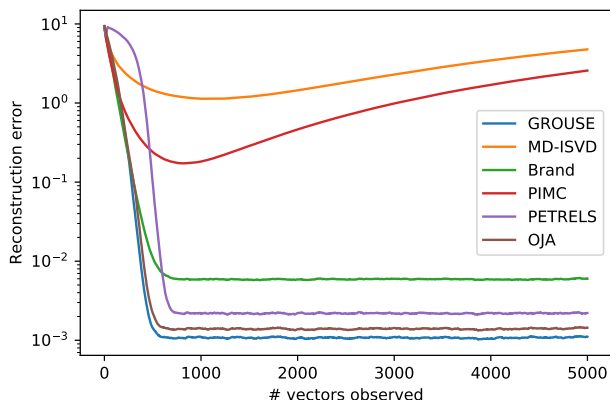

Fig. 6: Reconstruction error versus the number of observed snapshots with a subspace that slowly rotates over time.

experiments we set $\delta_0 = 10^{-5}$. Again, the noise level is set as $\sigma = 10^{-5}$ and the fraction of observation is set as $\alpha = 0.3$, but here we set c to be the all-ones loading vector. The results can be seen in Fig. 6; notice both that some of the algorithms fail to converge (again MD-ISVD and PIMC), and the rest converge but do not achieve as good accuracy as in the static case. This is to be expected since none of the algorithms is doing a prediction, but instead only tracking based on available data; therefore the constant difference between the true subspace at two consecutive time steps contributes to the noise floor for the algorithms.

VII. CONCLUDING REMARKS

In this paper, we have reviewed a variety of streaming PCA and subspace tracking algorithms, focusing on the ones that have a near-linear memory complexity and computational time, appealing convergence rates, and can tolerate missing data. Convergence guarantees based on classical and new asymptotic analysis as well as recent finite-sample non-asymptotic analysis are discussed. While we divide our discus-

sions intro algebraic and geometric approaches, it is important to point out that these approaches are not actually distinct. For example, Oja's is equivalent to Krasulina's, a stochastic gradient descent method, by ignoring second-order terms; GROUSE is equivalent to a version of Incremental SVD that is agnostic to past singular values [53]; and the PAST algorithm can also be interpreted from both perspectives [21]. It is an exciting open question to understand the connections between algebraic and geometric methods more generally.

Due to space limitations, we have focused on the problem of estimating a single low-dimensional subspace under the squared loss for streaming data with possibly missing entries, which is most suitable when the noise is modeled as Gaussian. There are many important extensions, motivated by real-world applications, including subspace tracking for non-Gaussian data [76], [77], tracking a union-of-subspace model [78], [79], tracking a low-dimensional subspace with multi-scale representations [80], subspace tracking in the presence of outliers and corruptions [27], [81]–[85], adaptive sampling [86], for an incomplete list. Interested readers are invited to go to the above cited references for details.

REFERENCES

[1] I. Jolliffe, *Principal Component Analysis.* Springer-Verlag, 1986.

[2] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Fifth International Conference on Computer and Information Science.* Citeseer, 2002, pp. 27–28.

[3] R. Kennedy, L. Balzano, S. J. Wright, and C. J. Taylor, "Online algorithms for factorization-based structure from motion," *Computer Vision and Image Understanding*, 2016.

[4] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[5] L. L. Scharf, "The SVD and reduced rank signal processing," *Signal processing*, vol. 25, no. 2, pp. 113–133, 1991.

[6] P. Comon and G. H. Golub, "Tracking a few extreme singular values and vectors in signal processing," *Proceedings of the IEEE*, vol. 78, no. 8, pp. 1327–1343, 1990.

[7] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.

[8] Y. Chen and Y. Chi, "Harnessing structures in big data via guaranteed low-rank matrix estimation," *arXiv preprint arXiv:1802.08397, IEEE Signal Processing Magazine*, Jul. 2018, to appear.

[9] P. Gao, M. Wang, S. G. Ghiocel, J. H. Chow, B. Fardanesh, and G. Stefopoulos, "Missing data recovery by exploiting low-dimensionality in power system synchrophasor measurements," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1006–1013, 2016.

[10] A. Majumdar, A. Gogna, and R. Ward, "A low-rank matrix recovery approach for energy efficient eeg acquisition for a wireless body area network," *Sensors*, vol. 14, no. 9, pp. 15 729–15 748, 2014.

[11] J. P. Delmas, "Subspace tracking for signal processing," *Adaptive Signal Processing: Next Generation Solutions*, pp. 211–270, 2010.

[12] E. J. Candès and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[13] D. L. Pimentel-Alarcón, N. Boston, and R. D. Nowak, "A characterization of deterministic sampling patterns for low-rank matrix completion," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 623–636, 2016.

[14] S. T. Smith, "Subspace tracking with full rank updates," in *Signals, Systems & Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on*, vol. 1. IEEE, 1997, pp. 793–797.

[15] M. Brand, "Incremental singular value decomposition of uncertain data with missing values," in *Proceedings of the 7th European Conference on Computer Vision-Part I*. Springer-Verlag, 2002, pp. 707–720.

[16] I. Karasalo, "Estimating the covariance matrix by signal subspace averaging," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 34, no. 1, pp. 8–12, 1986.

[17] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of mathematical biology*, vol. 15, no. 3, pp. 267–273, 1982.

[18] A. Balsubramani, S. Dasgupta, and Y. Freund, "The fast convergence of incremental PCA," in *Advances in Neural Information Processing Systems*, 2013, pp. 3174–3182.

[19] T. Krasulina, "The method of stochastic approximation for the determination of the least eigenvalue of a symmetrical matrix," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 6, pp. 189 – 195, 1969.

[20] K. Abed-Meraim, A. Chkeif, and Y. Hua, "Fast orthonormal PAST algorithm," *IEEE Signal processing letters*, vol. 7, no. 3, pp. 60–62, 2000.

[21] Y. Hua, Y. Xiang, T. Chen, K. Abed-Meraim, and Y. Miao, "A new look at the power method for fast subspace tracking," *Digital Signal Processing*, vol. 9, no. 4, pp. 297–314, 1999.

[22] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*. IEEE, 2010, pp. 704–711.

[23] Y. Chi, Y. C. Eldar, and R. Calderbank, "Petrels: Parallel subspace estimation and tracking by recursive least squares from partial observations," *IEEE Transactions on Signal Processing*, vol. 61, no. 23, pp. 5947–5959, 2013.

[24] H. Guo, C. Qiu, and N. Vaswani, "An online algorithm for separating sparse and low-dimensional signal sequences from their sum," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4284–4297, 2014.

[25] B. Yang, "Projection approximation subspace tracking," *IEEE Transactions on Signal processing*, vol. 43, no. 1, pp. 95–107, 1995.

[26] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Transactions on Signal Processing*, vol. 63, no. 10, pp. 2663–2677, 2015.

[27] J. Feng, H. Xu, and S. Yan, "Online robust PCA via stochastic optimization," in *Advances in Neural Information Processing Systems*, 2013, pp. 404–412.

[28] B. Yang, "Asymptotic convergence analysis of the projection approximation subspace tracking algorithms," *Signal processing*, vol. 50, no. 1, pp. 123–136, 1996.

[29] T. Chen, Y. Hua, and W.-Y. Yan, "Global convergence of Oja's subspace algorithm for principal component extraction," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 58–67, 1998.

[30] L. Ljung, "Analysis of recursive stochastic algorithms," *IEEE Transactions on Automatic Control*, vol. 22, no. 4, pp. 551–575, 1977.

[31] H. J. Kushner, "Convergence of recursive adaptive and identification procedures via weak convergence theory," *IEEE Trans. Automat. Contr.*, vol. 22, no. 6, pp. 921–930, Dec. 1977.

[32] C. Wang and Y. M. Lu, "Online learning for sparse pca in high dimensions: Exact dynamics and phase transitions," in *Information Theory Workshop (ITW), 2016 IEEE*. IEEE, 2016, pp. 186–190.

[33] C. Wang, Y. Eldar, and Y. M. Lu, "Subspace estimation from incomplete observations: A precise high-dimensional analysis," in *Signal Processing with Adaptive Sparse Structured Representations (SPARS) Workshop*, 2017.

[34] C. Wang, J. Mattingly, and Y. M. Lu, "Scaling limits: Exact and tractable analysis of online learning algorithms with applications to regularized regression and PCA," *arXiv:1712.04332 [cs.LG]*, 2017.

[35] C. Wang, Y. C. Eldar, and Y. M. Lu, "Subspace estimation from incomplete observations: A high-dimensional analysis," *arXiv preprint arXiv:1805.06834 [cs.LG]*, 2018.

[36] I. Mitliagkas, C. Caramanis, and P. Jain, "Memory limited, streaming PCA," in *Advances in Neural Information Processing Systems*, 2013, pp. 2886–2894.

[37] M. Hardt and E. Price, "The noisy power method: A meta algorithm with applications," in *Advances in Neural Information Processing Systems*, 2014, pp. 2861–2869.

[38] C. J. Li, M. Wang, H. Liu, and T. Zhang, "Near-optimal stochastic approximation for online principal component estimation," *Mathematical Programming*, vol. 167, no. 1, pp. 75–97, 2018.

[39] O. Shamir, "Convergence of stochastic gradient descent for PCA," in *International Conference on Machine Learning*, 2016, pp. 257–265.

[40] Z. Allen-Zhu and Y. Li, "First efficient convergence for streaming k-pca: A global, gap-free, and near-optimal rate," in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct 2017, pp. 487–492.

[41] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[42] E. Schmidt, "Zur theorie der linearen und nicht linearen integralgleichungen. i teil. entwicklung willkürlichen funkionen nach system vorgeschriebener." *Mathematische Annalen*, vol. 63, pp. 433–476, 1907.

[43] G. W. Stewart, "On the early history of the singular value decomposition," *SIAM review*, vol. 35, no. 4, pp. 551–566, 1993.

[44] G. Stewart, "Fredholm, Hilbert, Schmidt: Three fundamental papers on integral equations," 2011, available at www.cs.umd.edu/~stewart/FHS.pdf.

[45] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[46] R. Vershynin, "How close is the sample covariance matrix to the actual covariance matrix?" *Journal of Theoretical Probability*, vol. 25, no. 3, pp. 655–686, 2012.

[47] S. Muthukrishnan, "Data streams: Algorithms and applications," *Foundations and Trends® in Theoretical Computer Science*, vol. 1, no. 2, pp. 117–236, 2005.

[48] J. Bunch and C. Nielsen, "Updating the singular value decomposition," *Numerische Mathematik*, vol. 31, pp. 111–129, 1978.

[49] Y. Chi, "Nearest subspace classification with missing data," in *Signals, Systems and Computers, 2013 Asilomar Conference on*. IEEE, 2013, pp. 1667–1671.

[50] C. Yang, D. Robinson, and R. Vidal, "Sparse subspace clustering with missing entries," in *International Conference on Machine Learning*, 2015, pp. 2463–2472.

[51] G. H. Golub, "Some modified matrix eigenvalue problems," *Siam Review*, vol. 15, no. 2, pp. 318–334, 1973.

[52] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear algebra and its applications*, vol. 415, no. 1, pp. 20–30, 2006.

[53] L. Balzano and S. J. Wright, "On GROUSE and incremental SVD," in *IEEE Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2013.

[54] R. Kennedy, C. J. Taylor, and L. Balzano, "Online completion of ill-conditioned low-rank matrices," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*. IEEE, 2014, pp. 507–511.

[55] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *Journal of mathematical analysis and applications*, vol. 106, no. 1, pp. 69–84, 1985.

[56] M.-F. Balcan, S. S. Du, Y. Wang, and A. W. Yu, "An improved gap-dependency analysis of the noisy power method," in *Conference on Learning Theory*, 2016, pp. 284–309.

[57] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[58] L. Balzano and S. J. Wright, "Local convergence of an algorithm for subspace identification from partial data," *Foundations of Computational Mathematics*, vol. 15, no. 5, pp. 1279–1314, 2015.

[59] D. Zhang and L. Balzano, "Global convergence of a grassmannian gradient descent algorithm for subspace estimation," in *Artificial Intelligence and Statistics*, 2016, pp. 1460–1468.

[60] ——, "Convergence of a grassmannian gradient descent algorithm for subspace estimation from undersampled data," *arXiv preprint arXiv:1610.00199*, 2016.

[61] D. P. Bertsekas, "Incremental gradient, subgradient, and proximal methods for convex optimization: A survey," *Optimization for Machine Learning*, vol. 2010, no. 1-38, p. 3, 2011.

[62] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM journal on Matrix Analysis and Applications*, vol. 20, no. 2, pp. 303–353, 1998.

[63] A. Eftekhari, L. Balzano, D. Yang, and M. B. Wakin, "SNIPE for memory-limited pca from incomplete data," *arXiv preprint arXiv:1612.00904*, 2016.

[64] T. Gustafsson, "Instrumental variable subspace tracking using projection approximation," *IEEE transactions on signal processing*, vol. 46, no. 3, pp. 669–681, 1998.

[65] R. Badeau, B. David, and G. Richard, "Fast approximated power iteration subspace tracking," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2931–2941, 2005.

[66] I. M. Johnstone, "On the distribution of the largest eigenvalue in principal components analysis," *Annals of statistics*, pp. 295–327, 2001.

[67] A.-S. Sznitman, "Topics in propagation of chaos," in *Ecole d'Eté de Probabilités de Saint-Flour XIX — 1989*, ser. Lecture Notes in Mathematics, P.-L. Hennequin, Ed. Springer Berlin Heidelberg, 1991, no. 1464, pp. 165–251.

[68] P. Billingsley, *Convergence of probability measures*, 2nd ed., ser. Wiley series in probability and statistics. Probability and statistics section. New York: Wiley, 1999.

[69] P. Jain, C. Jin, S. M. Kakade, P. Netrapalli, and A. Sidford, "Streaming PCA: Matching matrix bernstein and near-optimal finite sample guarantees for Oja's algorithm," in *Conference on Learning Theory*, 2016, pp. 1147–1164.

[70] C.-L. Li, H.-T. Lin, and C.-J. Lu, "Rivalry of two families of algorithms for memory-restricted streaming PCA," in *Artificial Intelligence and Statistics*, 2016, pp. 473–481.

[71] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2980–2998, 2010.

[72] B. Recht, "A simpler approach to matrix completion," *Journal of Machine Learning Research*, vol. 12, no. Dec, pp. 3413–3430, 2011.

[73] V. Koltchinskii, K. Lounici, A. B. Tsybakov *et al.*, "Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion," *The Annals of Statistics*, vol. 39, no. 5, pp. 2302–2329, 2011.

[74] M. A. Davenport and J. Romberg, "An overview of low-rank matrix recovery from incomplete observations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 608–622, 2016.

[75] C. De Sa, C. Re, and K. Olukotun, "Global convergence of stochastic gradient descent for some non-convex matrix problems," in *International Conference on Machine Learning*, 2015, pp. 2332–2341.

[76] Y. Shen, M. Mardani, and G. B. Giannakis, "Online categorical subspace learning for sketching big data with misses," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4004–4018, 2017.

[77] L. Wang and Y. Chi, "Stochastic approximation and memory-limited subspace tracking for poisson streaming data," *IEEE Transactions on Signal Processing*, vol. 66, no. 4, pp. 1051–1064, Feb 2018.

[78] L. Balzano, A. Szlam, B. Recht, and R. Nowak, "K-subspaces with missing data," in *Statistical Signal Processing Workshop (SSP), 2012 IEEE*. IEEE, 2012, pp. 612–615.

[79] X. Jiang and R. Willett, "Online data thinning via multi-subspace tracking," *arXiv preprint arXiv:1609.03544*, 2016.

[80] Y. Xie, J. Huang, and R. Willett, "Change-point detection for high-dimensional time series with missing data," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 12–27, 2013.

[81] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy, "Robust PCA, subspace learning, and tracking," *IEEE Signal Processing Magazine*, Jul. 2018, to appear.

[82] J. Goes, T. Zhang, R. Arora, and G. Lerman, "Robust stochastic principal component analysis," in *Artificial Intelligence and Statistics*, 2014, pp. 266–274.

[83] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[84] P. Narayanamurthy and N. Vaswani, "Provable dynamic robust PCA or robust subspace tracking," *arXiv preprint arXiv:1705.08948*, 2017.

[85] J. Zhan, B. Lois, H. Guo, and N. Vaswani, "Online (and offline) robust PCA: Novel algorithms and performance guarantees," in *Artificial Intelligence and Statistics*, 2016, pp. 1488–1496.

[86] A. Krishnamurthy and A. Singh, "Low-rank matrix and tensor completion via adaptive sampling," in *Advances in Neural Information Processing Systems*, 2013, pp. 836–844.

Laura Balzano (S'05-M'13-SM'18) is an assistant professor of Electrical Engineering and Computer Science at the University of Michigan. She received her Ph.D. in Electrical and Computer Engineering from the University of Wisconsin, Madison in 2012, supported by a 3M Fellowship. At Wisconsin she received the ECE Best Dissertation Award. She received her M.S. in Electrical Engineering from the University of California, Los Angeles in 2007 and her B.S. in Electrical and Computer Engineering from Rice University in 2001. She is an Intel Early Career Faculty Honor Fellow and a 3M Non-tenured Faculty Awardee. Her main research focus is online learning and optimization for low-rank models, with high-dimensional data that are highly incomplete or corrupted. She works on applications in networks, environmental monitoring, and computer vision.
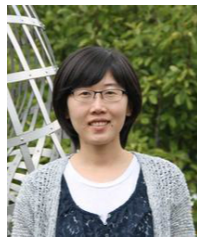
Yuejie Chi (S'09-M'12-SM'17) received the Ph.D. degree in Electrical Engineering from Princeton University in 2012, and the B.E. (Hon.) degree in Electrical Engineering from Tsinghua University, Beijing, China, in 2007. Since January 2018, she is an Associate Professor with the department of Electrical and Computer Engineering at Carnegie Mellon University, where she holds the Robert E. Doherty Early Career Professorship. She is the recipient of the IEEE Signal Processing Society Young Author Best Paper Award in 2013 and the Best Paper Award at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) in 2012. She received the NSF CAREER Award in 2017, Young Investigator Program Awards from AFOSR and ONR in 2015, Ralph E. Powe Junior Faculty Enhancement Award from Oak Ridge Associated Universities in 2014, Google Faculty Research Award in 2013, and Roberto Padovani scholarship from Qualcomm Inc. in 2010. She is an Elected Member of the MLSP and SPTM Technical Committees of the IEEE Signal Processing Society since January 2016. Her research interests include signal processing, machine learning, large-scale optimization, and their applications in data science, inverse problems, imaging, and sensing systems.
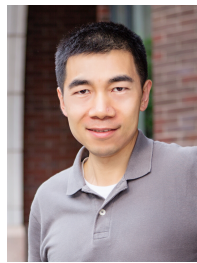
Yue M. Lu did his undergraduate studies at Shanghai Jiao Tong University, China. He then attended the University of Illinois at Urbana-Champaign, where he received the M.Sc. degree in mathematics and the Ph.D. degree in electrical engineering, both in 2007. Following his work as a postdoctoral researcher at the Audiovisual Communications Laboratory at Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, he joined Harvard University in 2010, where he is currently an Associate Professor of Electrical Engineering at the John A. Paulson School of Engineering and Applied Sciences.

He received the Most Innovative Paper Award of IEEE International Conference on Image Processing (ICIP) in 2006, the Best Student Paper Award of IEEE ICIP in 2007, and the Best Student Presentation Award at the 31st SIAM SEAS Conference in 2007. Student papers supervised and coauthored by him won the Best Student Paper Award of IEEE International Conference on Acoustics, Speech and Signal Processing in 2011, the Best Student Paper Award of IEEE Global Conference on Signal and Information Processing (GlobalSIP) in 2014, and the Best Student Paper (First Prize) of the IEEE CAMSAP Workshop in 2017. He is a recipient of the 2015 ECE Illinois Young Alumni Achievement Award.

He has been an Associate Editor of the IEEE Transactions on Image Processing, and he has served as an Elected Member of the IEEE Image, Video, and Multidimensional Signal Processing Technical Committee, and an Elected Member of the IEEE Signal Processing Theory and Methods Technical Committee. His research interests include theoretical and algorithmic aspects of high-dimensional statistical signal and information processing, imaging, and image processing.